ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD

Y COMPUTACIÓN

TESIS "CLASIFICACIÓN DE DATOS Y MANEJO DE INFORMACIÓN USANDO UN MODELO DE RED NEURONAL"

Trabajo previo a la obtención del titulo de:

Ingeniero en Electricidad

Especialización Computación

Autor:

JORGE CABELLO

Guayaquil — Ecuador

2003

AGRADECIMIENTOS

Al CRECE de la Escuela Superior Politécnica del Litoral (ESPOL) por su gentileza y colaboración en brindar información estadística para la realización de este proyecto.

Al Ingeniero Carlos Jordan, Profesor Director de La Tesis por su valiosa asistencia y dirección durante el desarrollo del presente proyecto.

DR. CRISTÓBAL MERA G.

Decano FIEC

ING. SERGIO FLORES

Miembro del Tribunal

ING. ENRIQUE PELÁEZ J.

Miembro del Tribunal

ING. CARLOS JORDÁN

Director de Tesis

DECLARACIÓN EXPRESA

"El patrimonio intelectual y la responsabilidad por los hechos, ideas y doctrinas expuestas en este proyecto, le corresponden exclusivamente a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

_____

Jorge Cabello Moran

ÍNDICE GENERAL

INTRODUCCIÓN

Todos sabemos que el cerebro humano supera en muchas tareas a los sistemas de cómputo digital, tareas tai como el reconocimiento de patrones. Una persona reconoce un patrón como una cara mucho más rápido y de manera sencilla que un computador, solo en las tareas matemáticas un ordenador supera a un cerebro humano.

Las redes neuronales intentan simular el procesamiento de un cerebro humano para aplicaciones donde la inteligencia humana no pueda ser emulada de forma satisfactoria por algoritmos aritméticos que pueden ser implementados en ordenadores. Estas redes neuronales deben cumplir características muy similares a las funciones del cerebro:

- Robustez: Mantener su funcionamiento de forma satisfactoria.
- Flexibilidad: Debe adaptarse a las nuevas circunstancias o situaciones.
- Inteligencia: Mantener su conocimiento mediante un autoaprendizaje.
- Consistencia: Debe lograr trabajar con información borrosa, incompleta, probabilística, con ruido o inconsistente.

- Integración: Debe mantener un comportamiento colectivo mediante Neuronas interconectadas.

[Tornado de Conceptos Fundamentales sobre las redes neuronales — Marcelo González]

Las Redes Neuronales Artificiales son aplicadas a un gran número de problemas reales de considerable complejidad, tales como reconocimiento de patrones, clasificación de datos, predicciones, etc. Su principal orientación es la solución de problemas que son demasiado complejos para las técnicas convencionales: problemas que no tienen un algoritmo específico para su solución, o cuyo algoritmo es demasiado complejo para ser encontrado.

Las Redes Neuronales Artificiales son aceptadas como nuevos sistemas muy eficaces para el tratamiento de la información en muchas disciplinas. La misión de esta tesis es establecer un material que brinde una buena base de conocimientos sobre la teoría de redes neuronales e implementar un modelo específico relacionado a un problema real.

CAPITULO No. 1

1. FACTOR P - PROBLEMÁTICA, CRITERIOS, VALORES Y ESTADÍSTICAS

1.1 PROBLEMÁTICA

La Escuela Superior Politécnica del Litoral, en su afán de establecer un mecanismo de pago de matriculas, que permita asignar valores más justos de acuerdo al nivel económico de los estudiantes, ha implantado una formula que contempla el promedio de calificaciones y el nivel socioeconómico del estudiante.

El nivel socioeconómico que se refleja en un valor o factor que permite a la Universidad evaluar los ingresos económicos del estudiante y su familia y el entorno social en el que se desenvuelve, se denomina el Factor P.

Este Factor P pondera a los estudiantes con valores que van de 3 para los estudiantes de nivel socioeconómico más bajo a 40 para los estudiantes de nivel socioeconómico más alto. Cada uno de los cinco criterios impactan de cierta manera al valor de este factor y estos valores asignados inciden directamente sobre el valor que el estudiante paga a la

Universidad por concepto de matricula para un determinado semestre. pero realmente este valor asignado es el adecuado?

Esta tesis presenta un estudio que permite establecer el impacto real de cada uno de los criterios que formula el factor P y entrega una herramienta que permite asignar de diferentes maneras estos criterios y ver la incidencia sobre el factor P. Por ejemplo, Si un estudiante recibe un factor P de 35, podemos evaluar cual es el impacto real sobre este valor por concepto de ingreso económico del grupo familiar del estudiante, institución de estudios primarios, ubicación domiciliaria, etc.

En caso de reasignar valores para el criterio de institución de estudios primarios, aun cuánto influye este valor al factor P?. En el factor P, que va de 3 a 40, realmente son estos los valores mínimos y máximos adecuados?. Preguntas como estas pueden ser respondidas con estudios que permitan pruebas para cada uno de los criterios y manejo de nuevos mínimos y máximos para el factor P, logrando observar el impacto real sobre los valores obtenidos con la formula original del Factor determinando así, silos valores que se asignan actualmente son los adecuados.

1.2 CRITERIOS PARA EL CÁLCULO DEL FACTOR P

El valor del factor P es calculado de acuerdo a la siguiente formula:

FORMULA P = COL + UBI + DEN + IPC + CEE

DONDE COL = Colegio de Procedencia

UBI = Ubicación de vivienda

DEN = Densidad Habitacional

IPC = Ingreso per capita del grupo familiar

CEE = Consumo per capita de energía eléctrica

[Tornado de la Universidad Politécnica del Litoral — CRECE]

Todas estas variables son valoradas de acuerdo a ciertos criterios, los cuales se describen a continuación:

COL Colegio de Procedencia

De acuerdo al colegio de procedencia del estudiante al momento de graduarse se valoran de acuerdo a las siguientes asignaciones:

- Rango: 1 — 12
- Si el colegio es fiscal COL = 1
- Si el colegio es particular y el estudiante hubiese tenido beca completa, entonces COL = 50% CL. del colegio.

- Si el estudiante hubiese tenido media beca, entonces COL = 75% COL del colegio.
- Para colegios particulares, si el pago de pensión mensual es:

  Menos de 100.000, entonces COL = 1

  Mas de 100.000 hasta 200.000, entonces COL = 2

  Mas de 200.000 hasta 300.000, entonces COL = 3

  Mas de 300.000 hasta 400.000, entonces COL = 4

  Mas de 400.000 hasta 500.000, entonces COL = 5

  Mas de 500.000 hasta 600.000, entonces COL = 6

  Mas de 600.000 hasta 700.000, entonces COL = 7

  Mas de 700.000 hasta 800.000, entonces COL = 8

  Mas de 800.000 hasta 900.000, entonces COL = 9

  Mas de 900.000 hasta 1.200.000, entonces COL = 10

  Mas de 1.200.000 hasta 1.500.000, entonces COL = 11

  Mas de 1.500.000, entonces COL = 12

UBI Ubicación de Vivienda.

De acuerdo a la ubicación domiciliaria del estudiante al momento de graduarse se valoran de acuerdo a las siguientes asignaciones:

- Rango: 0 — 14
- Si el grupo familiar reside en Guayaquil y pertenece a un sector:

  Popular SP, entonces UBI = 0

Residencial de Interés Social RS, entonces UBI = 2

Residencial Medio RM, entonces UBI = 6

Residencial Medio Alto RD, entonces UBI 10

Residencial Alto RA, entonces UBI = 12

Residencial Exclusivo RE, entonces UBI = 14

- Si el grupo familiar reside en Quito y pertenece a un sector:

  Popular SP, entonces UBI = 0

  Residencial Medio RM, entonces UBI = 6

  Residencial Alto RA, entonces UBI = 12

- Si el grupo familiar reside en Galápagos, entonces UBI = 2
- Si el grupo familiar reside en otra provincia, cabecera cantonal o sector rural y pertenece a un sector:

  Popular SP, entonces UBI = 0

  Residencial Medio RM, entonces UBI = 6

[Tornado de la Universidad Politécnica del Litoral — CRECE]

DEN Densidad habitacional

De acuerdo a la densidad habitacional actual del estudiante se valoran de acuerdo a las siguientes asignaciones:

- Rango:1—4
- Si de la división de los metros cuadrados de la vivienda para el número de miembros del grupo familiar es:

Menos de 18 m2, entonces DEN = 1

Mas de 18 m2 hasta 36 m2, entonces DEN = 2

Mas de 36 m2 hasta 54 m2, entonces DEN = 3

Mas de 54 m2, entonces DEN = 4

[Tornado de la Universidad Politecnica del Litoral — CRECE]

IPC Ingreso per capita del Grupo Familiar

De acuerdo al ingreso per capita del grupo familiar, el valor de IPC se calcula de acuerdo a:

- Rango:0—4
- Si los ingresos mensuales del grupo menos el valor de arriendo o hipoteca, dividido para el número de personas que conforma el grupo es:

  Mas de 100.000 hasta 400.000 sucres, entonces IPC = 0

  Más de 400.000 hasta 1.000.000 sucres, entonces IPC = 2

  Más de 1.000.000 hasta 2.000.000 sucres, entonces IPC = 3

  Más de 2.000.000, entonces IPC = 4

CEE Consumo de energía eléctrica

De acuerdo al consumo de energía eléctrica del grupo familiar, el valor de CEE se calcula de acuerdo a:

- Rango:1—6

- Si el promedio de los últimos 3 meses del valor per capita del consumo mensual de energía eléctrica es:

  Hasta 100.000 sucres, entonces CEE = 1

  Mas de 100.000 hasta 200.000, entonces CEE = 2

  Mas de 200.000 hasta 300.000, entonces CEE = 3

  Mas de 300.000 hasta 500.000, entonces CEE = 5 Mas de 500.000, entonces CEE = 6

[Tornado de la Universidad Politécnica del Litoral — CRECE]

Premisas generales

- Los estudiantes tomaran un valor P entre 3 y 40 puntos
- Los estudiantes extranjeros no residentes tendrán un factor P = 40
- Los estudiantes extranjeros residentes que no sean graduados a nivel superior se asignara un P igual a un estudiante nacional.
- Los estudiantes que ya posean titulo profesional, nacional o extranjero residente, se asigna un P = 25
- Los estudiantes que por convenios o decisión institucional son sujetos a PAGO MÍNIMO se asigna un valor P = 7

Ejemplo:

Un estudiante con las siguientes características:

- Proviene de un colegio con una pensión mensual de 840.000 sucres.
- Reside en Guayaquil, en la ciudadela Alborada del norte de la ciudad.
- El lugar donde habita tiene un área total de 150 mts^2 y conforman un grupo familiar de 4 personas.
- El grupo familiar tiene un ingreso per capita de 12 '000.000 de sucres y paga 4'OOO.OOO por concepto de arriendo.
- Tiene un egreso mensual promedio de 450.000 por concepto de consumo de energía eléctrica.

Obtiene el siguiente Factor P:

COL: 9 (Colegio Particular Y pensión 840.000)

UBI: 6 (Sector Residencial Medio)

DEN: 3 (Densidad = 150 mts^2/4 = 37.5 mis^2)

IPC: 3 (Ingreso = (12'OOO.OOO — 4 '000.000)/4 = 2'OOO.OOO)

CEE: 5 (Promedio mensual de consumo eléctrico = 450.000)

FACTORP = CCL+UBI+DEN+IPC+CEE

= 9+6+3+3+5

= 26

1.3 ESTADÍSTICAS

De acuerdo a una muestra tomada en todas las facultades de la ESPOL se muestra una distribución del Factor P de acuerdo a como se muestra en el grafico 1.3.1:

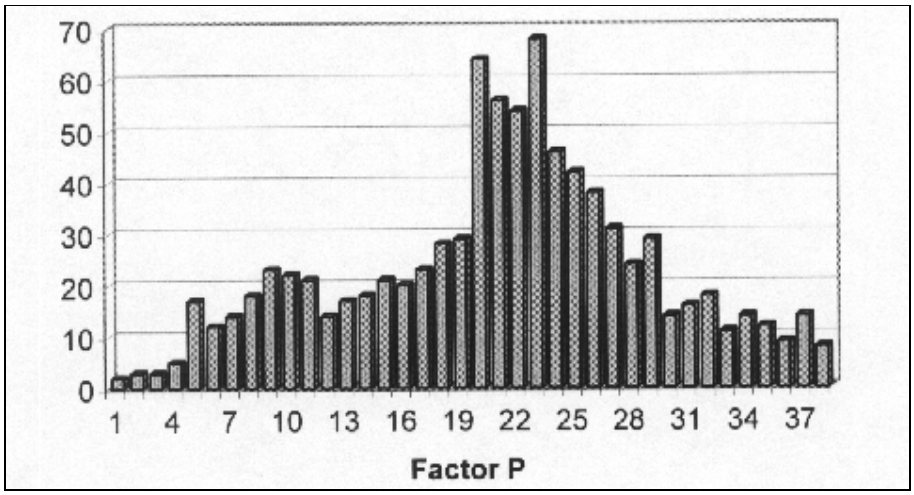


Figura 1.3.1

CAPÍTULO 2

2. TEORÍA BÁSICA

2.1. HISTORIA Y FUNDAMENTOS DE LAS REDES NEURONALES

Existe actualmente una tendencia a establecer un nuevo campo de las ciencias de la computación que integraría los diferentes métodos de resolución de problemas que no pueden ser descritos fácilmente mediante un enfoque algorítmico tradicional. Estos métodos, de una forma u otra, tienen su origen en la emulación, más o menos inteligente, del comportamiento de los sistemas biológicos.

Así, algunos autores se refieren a él como Computación Cognitiva (Cognitive Computing), otros lo denominan Computación del mundo real (Real-World Computing), y también se utiliza la expresión Computación Soft (Soft Computing), para distinguirlo del enfoque algorítmico tradicional que seria, en este caso, la Computación Hard, otro término alternativo seria el de Brainware. *[Tornado de Computer Based Training on Neural Nets. Richards Jackes]*

Con las Redes Neuronales se intentará expresar la solución de problemas complejos, no como una secuencia de pasos, sino como la evolución de unos sistemas de computación, inspirados en el funcionamiento del cerebro humano y dotados, por tanto, de una cierta "inteligencia"; los cuales, no son sino la combinación de una gran cantidad de elementos simples de proceso (neuronas) interconectados que, operando de forma masivamente y paralela, consiguen resolver problemas relacionados con el reconocimiento de formas o patrones, predicción, codificación, clasificación, control y optimización.

El concepto de una red neuronal no es nuevo. El objetivo de crear una red neuronal fue originalmente concebido como un intento por modelar la fisiología del cerebro. El objetivo era crear un modelo capaz de realizar procesos del pensamiento humano, inclusive muchos de los primeros trabajos fueron realizados por fisiólogos y psicólogos.

Uno de los primeros investigadores dedicados al estudio neuronal fue William James en 1890, cuya afirmación "La actividad de un punto en la corteza cerebral (neurona) es modelada por la suma ponderada de las entradas", sirvió de base para el trabajo realizado por M.S. McCulloch y W.A.Pitts en 1943 para modelar la actividad nerviosa del cerebro, el que hasta hoy sirve como fundamento para la construcción de muchas redes

neuronales. Posteriormente, Donald 0. Hebb en 1949 planteó las reglas fisiológicas de aprendizaje para la modificación de la sinapsis, donde se supone que se guarda la información.

[Tornado de Intmduction to a,tifidallntellegenCe, Jackson-Dover]

K.S. Lashley en 1950, formuló una representación distribuida de la memoria. Farley y Clark en 1954 simularon modelos para relaciones adaptivas de estimulo-respuesta. En 1958 F. Rosenblatt plantea el modelo Perceptron. Widrow y Hoff en 1960, Caianielo en 1961, y Steinbuch en 1961 elaboraron diversas teorías y modelos acerca de las neuronas. En esta época se realizaron numerosas implementaciones de computadores neuronales.

Se han logrado grandes avances merced a los aportes de T. Kohonen (1972) y J.A. Anderson (1972) sobre la autoorganización, y de muchos otros, entre los que destacan Hopfield (1982), Rumelhart y McClelland (1986), y Grossberg (1986).

Actualmente existen dos grandes tendencias en el estudio de las redes neuronales. Un primer grupo compuesto por los biólogos, físicos y psicólogos, trabaja en el desarrollo de modelos que imiten el comportamiento del cerebro.

El segundo grupo consiste de ingenieros que poseen el conocimiento de cómo estas neuronas artificiales pueden ser interconectadas para formar redes con poderosas capacidades computacionales. Estos últimos utilizan los modelos biológicos desarrollados por el primer grupo como punto de partida de sus investigaciones.

2.2. DEFINICIÓN, CARACTERÍSTICAS Y COMPONENTES DE UNA RED NEURONAL

2.2.1. Definición

Nuestro Universo entrega una diversidad de fuentes de entradas a nuestro cerebro, el cual las procesa de tal manera que se crean respuestas a estos estímulos. De forma muy organizada y sistemática, nuestro cerebro entreteje todos los estímulos posibles y los relaciona creando un sistema neuronal de información conformado por billones de neuronas, las cuales recepta la información externa y se encargan de ordenar al resto de nuestro cuerpo a realizar acciones.

Han sido muchos los intentos del hombre por comprender y construir una máquina que se asemeje a este sistema neuronal y realice tareas inteligentes.

Hace algunos años se han tratado de crear un sistema de inteligencia artificial que permita ejecutar tareas del mundo real naciendo así lo que conocemos como Redes Neuronales.

Entonces, una red neuronal es un grupo de elementos interconectados, unidades o nodos, los cuales funcionalmente están basados en una neurona animal. En otras palabras, una Red Neuronal artificial esta compuesta de un grupo de elementos de procesamiento altamente interconectados (Neuronas) trabajando al mismo tiempo para la solución de procesos específicos.

Es importante recalcar que las redes neuronales a igual que los humanos aprenden en base a la experiencia y entrenamiento. De esta combinación de conceptos nació una nueva forma de computación que trata de resolver problemas reales con soluciones automatizadas y fáciles de implementar.

Para comprender el funcionamiento y operación de los elementos o neuronas que conforman la red, hacemos una revisión funcional de una neurona biológica.

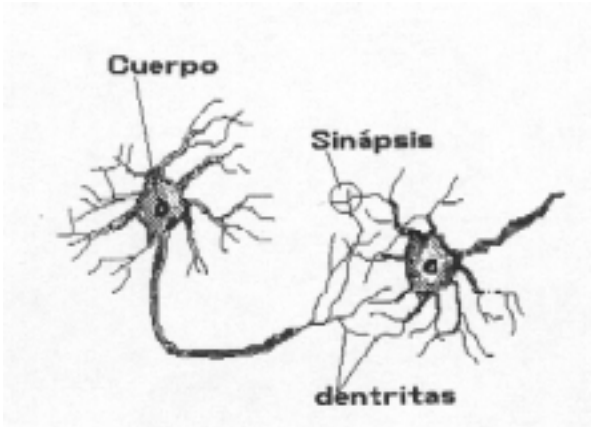


Figura 2.2.1

[Tornado de Computacional Neuroscience, Bower]

Como se aprecia en la figura 2.2.1, cada neurona en el cerebro está compuesta básicamente por un cuerpo, axones y dendritas. Las dendritas forman un “cepillo filamentoso” muy fino que rodea el cuerpo de la neurona. El axon puede considerarse como un tubo largo y fino que se subdivide en numerosas ramas que terminan en pequeños bulbos, los cuales tienen contacto con las dendritas de las otras células. La pequeña separación entre una terminación y una dendrita es llamada sinapsis. El axon de una neurona puede formar conexiones sinápticas con muchas otras neuronas.

Funcionalmente, las dendritas reciben señales desde otras células a través de los puntos de conexión Ilamados sinapsis. La fuerza de una conexión dada es determinada por la eficiencia de la transmisión sináptica. Desde ahí las señales son pasadas al cuerpo de la célula. Las señales que llegan de las dendritas pueden ser excitadoras o inhibitorias, y si la suma ponderada de éstas, realizada dentro del cuerpo de la neurona, supera su "umbral de activación" dentro de un tiempo suficiente, la neurona se disparará, enviando un impulso nervioso a través de su axon.

[Tornado de Cornputacional Neurosaence, Bower]

En comparación, una neurona artificial trata de imitar la funcionalidad de una neurona humana. Una red de neuronas consiste en varios de estos elementos (neuronas) trabajando juntos en la forma ya mencionada. Usualmente estos elementos se organizan en grupos conocidos con el nombre de capas. Una red típica consiste en una sucesión de capas conectadas entre ellas en forma total o aleatoria.

Estas redes poseen dos capas que tienen contacto con el exterior, que son la capa de entrada y la de salida. Las capas que no se conectan con el exterior reciben el nombre de capas ocultas. Cuando las señales aparecen estas son cuantificadas y multiplicadas por un peso, resultado el cual indica la ponderación de la estimulación. La suma de las

estimulaciones creadas por las señales de entrada produce un valor conocido como unidad de activación. Si esta unidad de activación excede un limite definido, la neurona se activa y emite una respuesta de salida. Podemos entonces asumir una neurona artificial que recibe *n* entradas o señales $x_1 + x_2 + \ldots + x_n$ y pesos $w_1 + w_2 + \ldots + w_n$. Debido al concepto binario, estas señales solo pueden tener una valor de "1" o "0". Entonces la activación de una neurona esta dada por la ecuación:

$$a = x_1w_1 + x_2w_2 + \ldots + x_nw_n = \sum_i^n x_n w_n$$

[Tornado de Redes neuronales, Algoritrnos y Técnicas de Prograrnadón, Freernan-Skapura]

Las Redes neuronales artificiales son modelos computarizados inspirados en la estructura a bajo nivel del cerebro. Se componen de grandes cantidades de unidades de procesamiento sencillas llamadas neuronas, conectadas por enlaces de varias fuerzas. Podemos citar una infinidad de frases que definan una red neuronal, entre otras:

Estudio de Redes Neuronales de DARPA: " Una red neuronal es un sistema compuesto de muchos elementos operando en paralelo, cuya función es determinada por la estructura de la red, fuerza de conexiones y el procesamiento realizado por los elementos computacionales de los nodos."

Haykin, S. (1994, Neural Networks: A comprehensive foundation, NY): “Una red neuronal es un procesamiento distribuido masivamente paralelo que tiene una tendencia natural para almacenar conocimiento empírico y hacerlo posible para el uso. Recuerda al cerebro en dos aspectos:

1.- Conocimiento se adquiere por la red a través de un proceso de aprendizaje
2.- Las conexiones interneurónicas se conocen como pesos sinápticos y se usan para almacenar el conocimiento.”

[Tornado de Computaciona/ Neuroscience, Bower]

[Tornado de An Introduction to Neural Nets, Anderson]

2.2.2. Características de una red neuronal.

Habiendo revisado los componentes que conforman una red neuronal y su funcionamiento, estamos en capacidad de diferenciar e identificar las principales características de una red artificial.

Una red neuronal es capaz de aprender. Las redes neuronales no ejecutan instrucciones secuénciales; así tampoco contienen memoria de almacenamiento, de instrucciones o de datos. En vez de esto, las redes

neuronales son entrenadas, presentándoles ejemplos de entradas y salidas, los que son memorizados alterando los vectores de pesos.

Tienen la capacidad de Generalizar, son capaces de acomodar los pesos de interconexión de modo de lograr una salida correcta frente a una entrada determinada. Esto sucede aún cuando una de las neuronas esté inhabilitada o alterada (tolerancia a fallas), lo que en la computación clásica puede ser logrado solo con algoritmos complejos y de alto costo.

Son capaces de Abstraer, siendo así un ente ideal desde un conjunto de entrenamiento no ideal, y recordar algo que no necesariamente se le haya enseñado. Son Veloces y cada neurona de la red es un procesador que opera sobre sus entradas independientemente de los otros procesadores y la convergencia ocupa a todas ellas, aunque se agreguen más procesadores. Esto contrasta con los problemas que presenta la programación paralela convencional.

Podemos concluir que el procesamiento es diferente al de una red humana, pues la combinación de las señales produce una nueva señal, en contraste a la ejecución de una instrucción guardaban en memoria.

La información se almacena una serie de pesos y no en un programa. Los pesos se adaptan a la red en un proceso de entrenamiento y aprendizaje.

No son susceptibles a ruido, pues cuando aparecen pequeños cambios en las señales de entrada no cambian drásticamente la señal de salida.

En cuanto al tamaño de las Redes neuronales, puede haber una o más capas ocultas entre las capas de entrada y salida. El tamaño de las redes depende del número de capas y del número de neuronas ocultas por capa. En la red el número de capas se define por si hay una o más capas de neuronas ocultas entre la entrada y la salida. El número de capas se cuenta a menudo a partir del número de capas de pesos (en vez de las capas de neuronas). El número de unidades ocultas está directamente relacionado con las capacidades de la red. Para que el comportamiento de la red sea correcto, se tiene que determinar apropiadamente el número de neuronas de la capa oculta.

La construcción de la red neuronal comprende 6 etapas comenzando con la conceptualizaciòn del modelo a usar. Se debe conocer claramente el modelo con sus entradas y salidas. Después de la

conceptualizaciòn del modelo, se debe hacer una recopilación de los datos. Los datos son procesados y formateados en una etapa de procesamiento, para que luego se cree la arquitectura del modelo. Se entrena la red con los valores mas indicados para el buen funcionamiento de la misma, y finalmente se analiza la red con valores reales.

El entrenamiento de una red es todo el proceso de aprendizaje que realiza una red neuronal. Su objetivo es lograr que la aplicación de un conjunto de entradas produzca el conjunto de salidas deseado (o uno al menos consistente). Dicho entrenamiento se realiza aplicando secuencialmente vectores de entrada, a la vez que se ajustan los pesos de la red de acuerdo a un procedimiento predeterminado, los cuales convergen gradualmente a valores tales que cada vector de entrada produce el vector de salida deseado. En cuanto a los tipos de aprendizaje, existen dos tipos: el aprendizaje supervisado y el no supervisado. Estos serán analizados en los capítulos posteriores, Modelos y Topologías.

2.3. Modelos y Topologías

Existe una gran variedad de tipo de modelos neuronales, los cuales se pueden clasificar dependiendo de su estructura, de su manera de aprender u operar y de su aplicación.

Podemos tener tres tipos de clasificaciones:

1. Clasificación por estructura.
2. Clasificación por aprendizaje.
3. Clasificación por aplicación.

En cuanto a la clasificación por estructura, nos concentramos en los componentes de la red, tales como, las capas, sus neuronas y conexión y tipos de funciones de activación. Entonces, encontraremos diversos modelos que se clasifican de acuerdo al número de sus componentes. Por ejemplo, si hablamos de redes neuronales de una capa, se las conoce como redes monocapas; y si hablamos de redes que poseen capas ocultas, hablamos de redes multicapas. Las redes monocapa solo cuentan con una capa de neuronas, que intercambian señales con el exterior y que constituyen a un tiempo la entrada y salida del sistema. En las redes monocapa, (red de Hopfield o red Brain-State-in-Box, máquina de Boltzman, máquina de Cauchy), se establecen conexiones laterales entre las neuronas, pudiendo existir, también conexiones auto recurrentes

(la salida de una neurona se conecta con su propia entrada), como en el caso del modelo Brain-State-in Box.

[Tornado de An Introduction to Neural Nets, Anderson]

Las redes multicapa disponen de conjuntos de neuronas jerarquizadas en distintos niveles o capas, con al menos una capa de entrada y otra de salida, y, eventualmente una o varias capas intermedias (ocultas).

Normalmente todas las neuronas de una capa reciben señales de otra capa anterior y envían señales a la capa posterior (en el sentido Entrada - Salida). A estas conexiones se las conoce como conexiones hacia delante o feedforward. Si una red solo dispone de conexiones de este tipo se la conoce como red feedforward. Sin embargo, puede haber redes en las que algunas de sus neuronas presenten conexiones con neuronas de capas anteriores, conexiones hacia atrás o feedback. En tal caso hablaremos de una red feedback o interactiva.

Entre las primeras destacan los distintos modelos de Kohonen, aunque presentan conexiones laterales y autorrecurrrentes, el Perceptron (multicapa) o M.L.P., las redes Adaline y Madaline, la Memoria Lineal Adaptiva y las Backpropagation. Entre las segundas debemos mencionar el

Cognitron y el Neocognitron, junto con los modelos de Resonancia y las máquinas multicapa de Boltzman y Cauchy.

[Tornado de An Introduction to Neural Nets, Anderson]

Dentro de esta clasificación la red neuronal más popular se la conoce como Modelo de red Multicapa FeedForward. El concepto de FeedForward quiere decir libre de ciclos, esto significa que las conexiones permitidas son entre las capas de entrada y la primera capa oculta, de la primera capa oculta a la segunda,. . .y de la última capa oculta a la de salida. La primera capa es llamada capa de entrada y recibe señales del mundo externo. La última capa es llamada capa de salida y propaga señales al mundo exterior.

Las otras capas son llamadas ocultas porque no se acceden directamente desde el mundo exterior ya que están envueltas exclusivamente en decisiones. El comportamiento de la red se basa en lo siguiente: Primero, las activaciones de las unidades de entrada son valuadas de acuerdo al problema que se quiere resolver. Después, las señales se propagan de la capa de entrada a la siguiente, con cada unidad realizando una suma ponderada de sus entradas y aplicando su función de activación. El proceso se repite hasta que la capa de salida se alcance. Las

activaciones de la salida representan la respuesta de la red a las entradas que recibe.

La red debe ser capaz de interpretar los valores de entrada y salida, lo cual depende de la aplicación por la que fue diseñada la red. Por ejemplo, si la red se pretende que prediga valores de mercado, las entradas pueden ser varias medidas financieras como inflación, precios, moneda, etc. Y las salidas pueden representar la predicción de la red respecto a los precios de mañana. Si la red predice una falla en un aeroplano, las entradas pueden ser la temperatura, nivel de aceite, altitud y edad del avión, con la salida siendo un valor entre 0 (para un estado tranquilo) y 1 (para un estado de pánico). En general, cualquier número de entradas y cualquier número de salidas puede ser especificado.

Dentro de la clasificación de redes multicapas debemos revisar el concepto de Perceptron. Rosenblatt diseño el percentron. Este contiene tres tipos de neuronas: sensoriales, asociativas y de respuesta. Las sensoriales tomaban entradas de fuera de la red, las unidades de respuestas propagan señales afuera de la red al mundo externo, y las asociativas son meramente internas. Rosenblatt desarrollo métodos para alterar los niveles sinápticos de forma que la red aprendiera a reconocer

clases de entradas. Por ejemplo, produjo una red que aprendió a responder a líneas verticales, pero no a horizontales.

El concepto de perceptron ha sido uno de los desarrollos más interesantes en el campo de reconocimiento de patrones. Los elementos de esta red son capaces de reconocer y clasificar patrones y además su disposición tiene grandes reminiscencias de las redes neuronales biológicas. El perceptron de capa simple es capaz de localizar la frontera de decisión lineal entre patrones pertenecientes a dos clases, siempre y cuando se garantice la separabilidad lineal de las dos clases de patrones.

[Tornado de Neural Networks for Pattern Recognition, Bishop]

En cuanto a la clasificación por aprendizaje, la parte más importante de la modulación de la red neuronal es determinar formas para ajustar los pesos de forma que la red realice un buen mapeo entre entradas y salidas: a esto se le llama “entrenamiento” de la red.

El mapeo de redes neuronales es tan complejo que aun cuando una red ha sido bien entrenada para resolver un problema, el usuario puede ser incapaz de entender como la red logra esto. Lo que se requiere son métodos de entrenamiento automático para el mapeo de entrada — salida. Un número de muestras son presentadas a la red y un

procedimiento de entrenamiento es usado para ajustar los pesos para que aprenda el mapeo. Las muestras de entrenamiento consisten de un conjunto de entradas juntas con la salida esperada.

La clasificación según el aprendizaje y operación básicamente se concentra en el algoritmo de aprendizaje que se utilice, tipo de supervisión que posea y en como opera la red en su evocación. En el siguiente tema de este capitulo se incluye una descripción completa de diferentes algoritmos de aprendizaje, con el fin de complementar y ayudar al diseñador en la creación de alguna aplicación.

Por último, su clasificación por aplicación puede ser enfocada de variadas formas: puede referirse a si la red es hetero-asociativa (caso en que el vector de entrada es asociado con un vector de salida de distinto tipo) *0* auto-asociativa (caso en que la entrada es asociada a un vector igual en la salida). Por otro lado, la aplicación también puede ser enfocada hacia la función que le asigne el diseñador a la red (predicción, clasificación, asociación, filtraje, conceptualizaciòn y optimizaciòn).

[Tornado de Neural Networks for Pattern Recognition, Bishop]

2.4. Entrenamiento y Aprendizaje

La capacidad de clasificación de la red neuronal depende del valor de los pesos sinápticos, que pueden ser preestablecidos o entrenados mediante mecanismos o técnicas de aprendizaje. Las redes neuronales se pueden clasificar en dos grandes grupos, los modelos de aprendizaje supervisado y no supervisado.

En el aprendizaje supervisado, hay un profesor que en la fase de entrenamiento le dice a la red como debe comportarse o cual debe de ser el comportamiento correcto. Las redes de entrenamiento supervisado han sido los modelos de redes más desarrollados desde inicios de estos diseños. Los datos para el entrenamiento están constituidos por varios pares de patrones de entrenamiento de entrada y salida.

En el aprendizaje no supervisado, solo se muestran los datos, se hallan algunas de sus propiedades del conjunto de datos y aprende a reflejar esas propiedades en la salida. La red debe aprender a reconocer, dependiendo del modelo particular de la red y del método de aprendizaje. Para este modelo de entrenamiento no supervisado, el conjunto de datos de entrenamiento consiste solo en los patrones de entradas. Por lo tanto, la red es entrenada sin el beneficio del maestro. La red aprende a

adaptarse basada en las experiencias recogidas de los patrones de entrenamiento anteriores. De acuerdo a esta clasificación, según el tipo de aprendizaje podemos distinguir cuatro tipos de redes;

1. Redes con aprendizaje no supervisado y evocación retroalimentada.
2. Redes con aprendizaje no supervisado y evocación hacia adelante
3. Redes con aprendizaje supervisado y evocación retroalimentada.
4. Redes con aprendizaje supervisado y evocación hacia adelante.

2.4.1. Redes neuronales con aprendizaje supervisado y evocación hacia adelante.

Dentro de esta clasificación podemos encontrar redes tipo Perceptron, Backpropagation, Adaline y Madaline.

Adaline

Este tipo de red es implementada en base a un tipo de neurona denominada Adaline. El esquema básico de un Adaline corresponde a una función de activación lineal, de la forma F(Zj) = Zj.

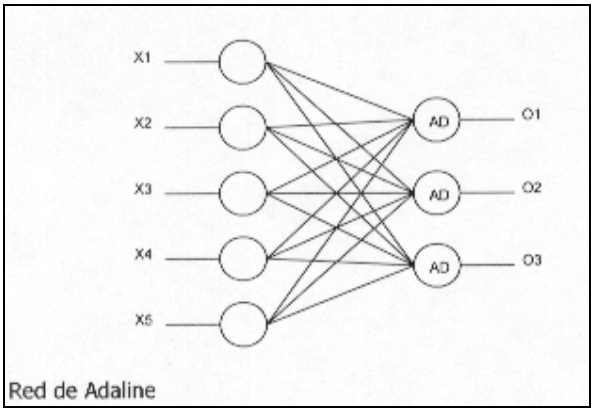


Figura 2.4.1

Dado que un Adaline es un modelo lineal, si se interconectan capas de Adaline no se tendrá una mayor eficiencia computacional debido a que una combinación lineal de estas unidades puede ser llevada a cabo por una sola unidad lineal. Por lo tanto, cuando se hable de una red de Adaline se entenderá que corresponde a una red de una sola capa.

El entrenamiento que posee esta red es del tipo supervisado, por lo que cada entrada que se presente debe ir acompañada de la salida que se desea. Las entradas son ponderadas y luego sumadas, lo que produce una salida que se compara con la deseada, originando un error. Este error es el que se utilizará en el ajuste de los pesos según el algoritmo de aprendizaje.

Inicialmente todos los pesos deben poseer valores aleatorios. Si todos los pesos iniciales son iguales el procedimiento de adaptación podría sumar o restar siempre la misma cantidad, y así caer en un mínimo local en el espacio de los pesos. El espacio de los pesos es un espacio Euclidiano de dimensión igual al número total de pesos; cada punto en este espacio es representado como un vector con los valores de los pesos.

Madaline (Múltiple Adaline)

Este tipo de red está formada por varias Adaline en paralelo, interconectadas en forma total con la capa de entrada, y seguidas cada una de ellas por una función de umbral del tipo signo. Posee una sola neurona de salida, denominada Madaline, la que realiza la función de entregar el valor que corresponda a la mayoría de sus entradas.

Como este tipo de red tiene una única salida binaria, ésta puede ser usada solo para discriminar entre dos clases. Si se desea discriminar entre más de dos clases, varias redes de Madaline independientes pueden ser usadas, una para cada par de clases.

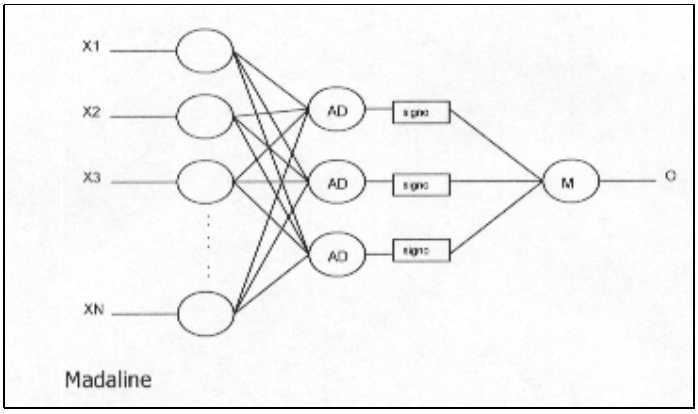


Figura 2.4.2

Si se denota por Yk la salida de la neurona k de la capa de las Adaline, entonces la salida de la red, 0, corresponde a:

$$O = \begin{cases} -1, siS\langle 0 \\ +1, siS \geq 0 \end{cases}$$

Donde

$$S = \sum_{k=o}^{m} Y_k$$

$$Y_K = \sum_{i=1}^{n} X_i * W_{i,k} + W_{0,k}$$

[Tornado de Elements of Artificial Neural Networks, Tarbe-Kulkami]

Cuando un patrón de entrada y su salida deseada son presentados a la red, ésta genera una salida que puede coincidir con la deseada, en cuyo caso no se realiza el algoritmo de aprendizaje. En caso contrario, se lleva a cabo el algoritmo de aprendizaje. El problema en este último caso

es determinar el error para adaptar los pesos de las Adaline, ya que no se especifica cual es su salida deseada. Durante el entrenamiento el patrón de entrada y la salida deseada son presentados a la red (entrenamiento supervisado). El aprendizaje es realizado solo por las neuronas Adaline, las que usan un algoritmo de aprendizaje similar al utilizado para una red Adaline.

Perceptron

Este sistema adaptivo, sugerido por Rosenblatt, es una combinación de diferentes unidades, de las cuales la primera capa (S) simplemente incluye los sensores del ambiente. Las señales producidas por los sensores son combinadas en la segunda capa de unidades llamadas “Elementos Asociativos” (A).

Esta capa está conectada en forma total o aleatoria a la primera y opera decodificando los patrones de entrada de modo de detectar sus características especificas: una respuesta activa se obtendrá si y solo si se da la combinación binaria del valor de la señal de entrada.

Según esto, los “Elementos Asociativos” van a actuar como un Adaline con pesos fijos y con su salida conectada a una función de transferencia del tipo umbral, que entrega valores lógicos 0 y +1.

La tercera capa, de los elementos de respuesta (R), constituye el propio sistema de aprendizaje: las conexiones de A a R son realizadas a través de conexiones variables, similares a las del Adaline. La figura presenta el esquema básico de una red Perceptron:

[Tornado de Neural Networks for Pattern Recognition, Bishop]

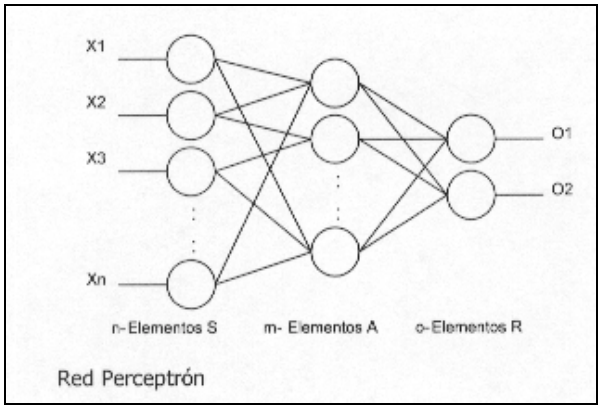


Figura 2.4.3

La diferencia más notable entre las funciones adaptivas del Perceptron y el Adaline es que cada elemento R es un disparador de umbral; éste posee solo dos estados de salida, que dependen de la entrada y de una función discriminante.

Las células asociativas (elementos A) son inicializadas con pesos fijos, cuyos valores pueden ser +1 0 -1. Su conexión con las células de respuesta (elementos R) es en forma aleatoria.

El proceso de aprendizaje es supervisado en los elementos R y puede tomar diferentes formas. La red puede ser utilizada tanto para casos discretos binarios como para continuos, pero en este ùltimo no se entrará en detalle debido a que existe otra red similar que opera en mejor forma y que se presentará mas adelante. El proceso de entrenamiento es prácticamente igual que en el caso del Adaline.

Retroprpagación (Backpropagation)

Una red perceptron es capaz de entrenar sus unidades de salida para aprender a clasificar los patrones de entrada linealmente separables. Para el caso de patrones no linealmente separables el proceso se hace más complejo y solo puede ser solucionado con redes multicapas. Sin embargo, si la salida posee error, existirá el problema de como determinar cual elemento de proceso o interconexión ajustar.

Una red de retropropagaciòn es capaz de resolver esto asignando a todos los elementos de proceso una cierta “responsabilidad” por la respuesta errónea. Esto se traduce en un error que es propagado desde la capa de salida a las neuronas de las capas previas de modo que éstas adapten sus pesos para minimizar dicho error y alcanzar el valor deseado.

Técnicamente, retropropagaciòn es una Ley de aprendizaje específica. Este término es usualmente usado para referirse a una arquitectura de red jerárquica que usa el algoritmo de retropropagaciòn para ajustar los pesos de interconexión de cada neurona de la red, basados en el error presente en la salida.

Esta red es una extensión de la estructura del Madaline con la diferencia de que a la salida de cada neurona existe una función no lineal y todos sus pesos son adaptables.

Dicha función no lineal debe ser continua, diferenciable, monotònicamente creciente y asintótica a un valor. En este trabajo utilizaremos específicamente dos tipos de estas funciones: la función sigmoidal y la tangente hiperbólica.

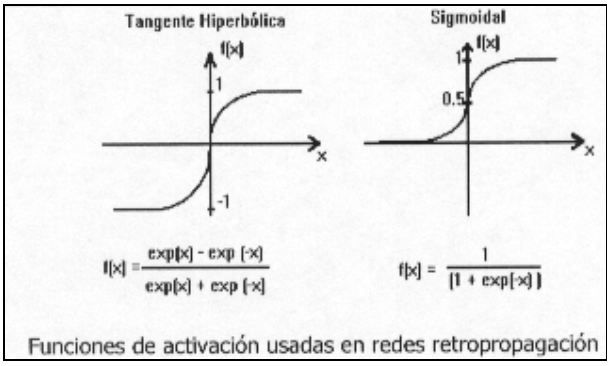


Figura 2.4.4

La estructura de esta red consta siempre de una capa de entrada, una de salida y al menos una capa oculta. Cada capa está totalmente conectada a la siguiente y no existe interconexión entre las neuronas de una misma capa. Los patrones de entrada y salida no tienen que ser necesariamente valores binarios, éstos pueden tomar cualquier valor que sea capaz de representar la red.

Muchas veces, cuando se utiliza una de las funciones de activación previamente mencionadas, la salida es generalmente escalada a los valores máximos y mínimos que logran dichas funciones.

Teóricamente no está limitado el número de capas ocultas, pero típicamente no son más de dos. Existen trabajos en que se han utilizado hasta tres capas ocultas, de modo de solucionar problemas complejos de clasificación de patrones. El número de neuronas en esta capa dependerá del diseñador de la red. Demasiadas neuronas en las capas ocultas harán que la red tenga dificultades al procesar nuevos tipos de patrones de entrada, es decir, hará generalizaciones. Pocas neuronas no permitirán a la red hacer suficientes representaciones internas de modo de generar sus mapas de entrada y salida.

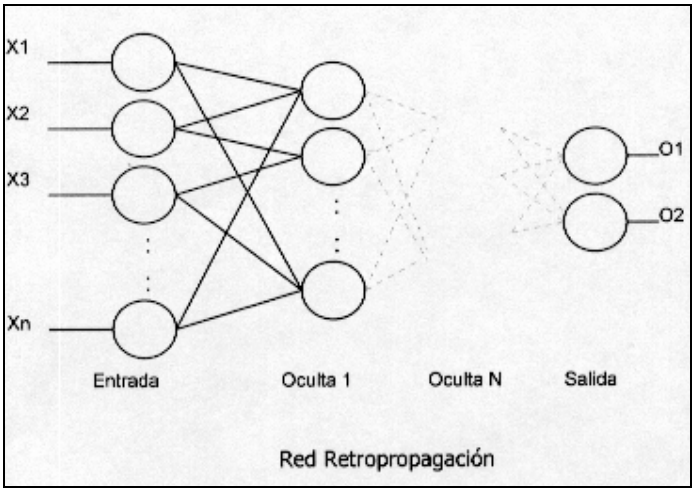


Figura 2.4.5

Todas las neuronas dentro de la red son idénticas al Adaline pero, como se mencionó anteriormente, con una función no lineal a la salida.

Como se mencionó anteriormente, el proceso de ajuste de los pesos se lleva a cabo primero en la capa de salida y posteriormente en las capas internas o escondidas, procediendo desde la más cercana a la salida a la más cercana a la entrada. El ajuste se realiza basándose en el algoritmo del descenso del gradiente o regla delta generalizada. Lo esencial de este procedimiento es que los términos de error requeridos para adaptar los pesos son retropropagados desde los nodos en la capa de salida hacia los nodos de las capas internas, de ahí que se le conozca como retropropagación. La aplicación de la regla delta implica dos fases:

Durante la primera fase la entrada es presentada y propagada hacia adelante a través de la red para computar el valor de salida 0 para cada unidad. Esta salida es luego comparada con el objetivo, resultando en una señal de error δ para cada unidad.

La segunda fase involucra un paso hacia atrás en la red, similar al paso hacia adelante, durante el cual la señal de error es entregada a cada unidad en la red realizándose los cambios apropiados en los pesos.

2.4.2. Redes con aprendizaje no supervisado y evocación hacia adelante.

Propagación por conteo hacia delante

(Counterpropagation)

Esta red basa su funcionamiento en la habilidad de aprender un mapa matemático por adaptación, en respuesta a ejemplos de una función de mapa.

De esta función se genera un conjunto de ejemplos (X1,Y1), (X2,Y2),..., donde X es una variable independiente escogida de acuerdo a la densidad fija de probabilidad ρ . Con estos ejemplos, se define

estadísticamente la relación deseada de entrada/salida, aprendiendo a través de ellos el mapeo matemático de la función.

Estas redes que siguen el procedimiento anterior son llamadas redes neuronales de mapeo, y dentro de esta clasificación se encuentra la red Propagación por Conteo. Esta arquitectura propuesta por Robert HechtNielsen (1987), tiene capacidades que la convierten en una red muy atractiva para una amplia gama de aplicaciones.

[Tornado de Computer Training on Neural Networks,Jackes]

Su arquitectura está formada por tres capas: Una capa de entrada, que contiene n unidades de dispersión que simplemente multiplexan las señales de entrada X1, X2,. . .,Xn.

Una capa intermedia llamada de Kohonen con N elementos de proceso que tienen señales de salida Z1, Z2, ..., ZN.

Una capa de salida llamada de Grossberg con m elementos de proceso que tienen señales Y'l, Y'2, ..., Y'm. Las salidas de la capa de Grossberg corresponden a las más importantes, debido a que representan aproximaciones a las componentes Y1, Y2, . . .,Ym.

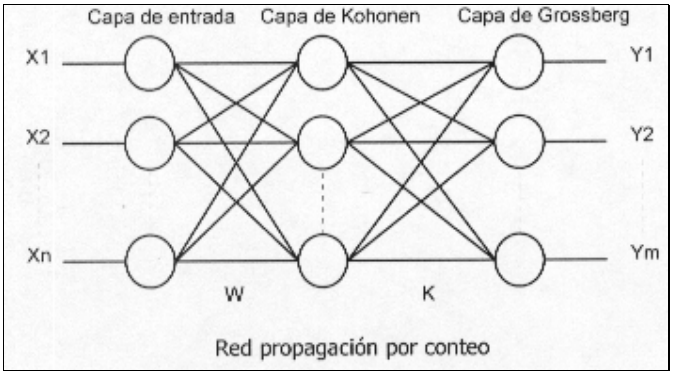


Figura 2.4.6

Los pesos de las capas de Kohonen y Grossberg inicialmente son asignados en forma aleatoria con valores convenientemente entre 0 y 1. El entrenamiento de la red se realiza en dos fases y por dos algoritmos diferentes.

Fase 1 del entrenamiento

Durante esta fase se aplica una secuencia de vectores de entrenamiento X y un algoritmo conocido como Aprendizaje de Kohonen, para ajustar los pesos Wi. El aprendizaje de Kohonen es un algoritmo autoorganizativo que opera en modo no supervisado, esto es, solo vectores de entrada (sin el vector de salida objetivo) son aplicados durante el entrenamiento y los pesos W son ajustados de manera tal que una neurona o grupo de ellas esté activo en la segunda capa. Lo que interesa de esta fase del entrenamiento es que se asegure la separación

de los vectores disímiles. El algoritmo de aprendizaje utilizado para el ajuste de los pesos queda determinado por:

a) Aplicar un vector de entrada X.

b) Encontrar el producto punto del vector X con cada conjunto de pesos conectados a cada neurona. Si Wi es el conjunto de pesos asociados con la neurona I de la capa 2, entonces para cada I, (i=1,...N) calcular el producto punto X.Wi.

c) Encontrar la neurona con el producto punto más alto y rotular esta neurona con la letra C.

d) Ajustar el vector de pesos asociado con la neurona C de acuerdo a la siguiente formula:

$$Wc(t + 1) = Wc(t) + \propto(X - Wc(t))Zc$$

$$Zc = \begin{cases} 1 \\ 0 \end{cases}$$ 1 si C es el menor índice para el cual:

$$||Wc(t) - X|| \leq ||Wj(t) - X||$$ para todo j.

[Tornado de Unsupervised Nt Kohonen 's Se/f Organizing Feature Map, Niebur]

e) Repetir los pasos 1 al 4 tantas veces como sea necesario para entrenar el sistema.

Al comienzo del entrenamiento, usualmente $\propto$ parte con un valor entre 0.5 y 0.8. Al progresar el entrenamiento, este valor decrece a 0.1 o menos.

Una vez que la capa de Kohonen se ha estabilizado (en otras palabras, los vectores se han congelado, después de alcanzar la equiprobabilidad), la capa de Grossberg comienza a aprender las salidas correctas para cada vector Wi de la capa de Kohonen. Esto se realiza en la segunda fase del entrenamiento.

Fase 2 del entrenamiento

Durante esta fase, se entrenan los pesos que conectan las capas 2 y 3 mediante el aprendizaje "outstar" de Grossberg. Para esto, se entrega un vector de entrada X y un vector de salida Y (objetivo), por lo que este método corresponde a un entrenamiento supervisado. El ajuste de los vectores Ki se realiza de la siguiente forma:

a) Entrar un vector X y un correspondiente vector objetivo Y.

b) Calcular el producto punto de cada vector de pesos Wi con el vector X. Encontrar el mayor de estos productos y rotular la neurona que corresponda en la capa de Kohonen con la letra C. Colocar su señal de salida en uno y la señal de salida de las otras neuronas de esta capa en cero.

c) Ajustar cada uno de los pesos Kcj, entre la neurona C de la capa de Kohonen y todas las neuronas de la capa de Grossberg, de acuerdo a la siguiente regla:

$$K_{cj}(t+1) = k_{cj}(t) + (dY_j - cK_{cj}(t))Z_c \qquad \text{, j=1,...,m}$$

donde c y d son constantes menores que 1 y que son reducidas durante el entrenamiento y Z es el nivel de señal de salida de la unidad C.

d) Repetir los pasos (1) al (4) tantas veces como sea necesario para entrenar el sistema.

Una vez que el entrenamiento ha terminado se establece el modo normal de operación. En este modo los vectores de entrada son presentados a la primera capa, para luego ser multiplicados por los pesos de las neuronas de Kohonen.

Una competencia se lleva a cabo, ganando aquellas unidades cuyos pesos Wi estén más cerca del vector de entrada X. El valor de la salida de la neurona i de la capa de Kohonen, queda determinada por:

$$Z_i = \sum_{j=0}^{n} W_{ij} - X_j$$

[Tornado de Kohonen Maps, Kaski]

La competencia puede establecer una única unidad ganadora o a un grupo de ellas. Las unidades ganadoras son puestas en uno y las no vencedoras en cero.

2.4.3. Redes con aprendizaje no supervisado y evocación retroalimentada.

Hopfield

La red de Hopfield es normalmente usada con entradas binarias. Esta red es la más apropiada cuando las representaciones binarias exactas son posibles, como en las imágenes en blanco y negro donde los píxeles son los valores de entrada. Esta red es menos apropiada cuando los valores de entrada son continuos, debido a que el problema fundamental de representación debe ser direccionado para convertir las cantidades análogas en valores binarios. Esta red, mostrada en la figura posee N nodos que contienen funciones de activación del tipo signo con entradas y salidas que toman valores +1 0 -1. La salida de cada neurona es realimentada a todos los otros nodos, excepto a si misma, con pesos denotados por Wij.

Las neuronas dentro de la red están basadas en el esquema de la Adaline, con la diferencia de que utiliza en su salida una función del tipo signo en vez de una lineal. La figura presenta un elemento de proceso de la red de Hopfield:

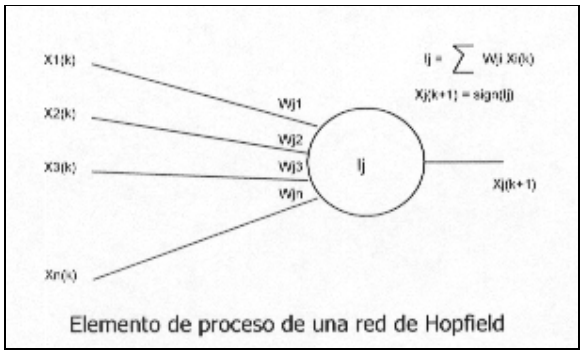


Figura 2.4.7

[Tornado de Neural Networks for Pattern Recognition, Bishop]

Básicamente, la asignación de los pesos corresponde al aprendizaje de la red. Estos son asignados con valores que dependerán de las clases de los patrones de entrada.

A pesar de la simplicidad del algoritmo, éste presenta dos graves limitaciones cuando es usado como una memoria asociativa. La primera Limitación es en cuanto al número de patrones que pueden ser almacenados en forma precisa. Si demasiados patrones son almacenados, la red puede converger a valores falsos. Hopfield demostró que el número apropiado de patrones que es capaz de almacenar tiene que ser menor

que 0.15 N, donde N es el número de nodos de la red. La segunda Limitación se refiere a que un ejemplar puede hacer inestable la red si su forma contiene demasiados bits en común con otros patrones. Se dice que el sistema es inestable cuando el patrón de salida converge a algùn otro ejemplar que no es el correcto.

Hamming

La red de Hamming, usada como red neuronal, es un clasificador de mínimo error para vectores binarios, donde el error es definido usando la distancia de Hamming.

En un clasificador de error mínimo, las clases son definidas por el significado de los vectores de ejemplo; los vectores de entrada son asignados a la clase para la cual la distancia entre el vector de ejemplo y el vector de entrada son mínimos.

La distancia de Hamming es el valor de la distancia entre dos vectores binarios y es definida como el número de bits en un vector de entrada que no sean los mismos que los bits de uno de los vectores de ejemplo.

Consideremos, por ejemplo, el siguiente conjunto de vectores con que la red ha sido entrenada:

C1 = [1,1,1,1,1,1]

C2 = [1,1,1,-1,-1,-1]

C3 = [-1,-1,-1,1,1,1]

C4= [-1,-1,-1,-1,-1,-1]

y sea el vector de entrada Cx = [1,-1,1,-1,-1,1]

Este vector de entrada Cx es comparado con cada uno de los vectores Cn, con n = 1, 2, 3, 4, resultando ser C2 el vector con la menor distancia según Hamming (posee solo dos bits distintos).

[Tornado de Neural Networks for Pattern Recognition, Bishop]

Una diferencia importante que posee esta red con la de Hopfield, mencionada es que, desde el punto de vista de la precisión y capacidad, la red de Hamming clasifica en mejor forma los vectores de entrada cuando los errores en los bits son independientes y aleatorios.

Por otro lado, la capacidad de almacenamiento que posee la red de Hamming es notablemente mayor. Por ejemplo: una red de Hopfield con 100 nodos puede almacenar en forma Optima alrededor de 10 patrones

de entrada y, en cambio, una red de Hamming puede almacenar cerca de 62 patrones bajo las mismas condiciones.

Estructuralmente, la red de Hamming está compuesta por dos redes: una primera que calcula las N distancias mínimas de Hamming para los M patrones de ejemplo, y una segunda red que selecciona el nodo con salida máxima.

La primera red está compuesta por dos capas: la primera es una capa de entrada que distribuye todas las entradas a la capa siguiente, llamada capa de formas. Esta última está formada por neuronas del tipo Adaline más una función del tipo signo en su salida. Las entradas a la red son del tipo binario, tomando los valores +1 0 -1. La segunda red tiene la misma estructura que una red de Hopfield pero la asignación de los pesos es distinta y sus entradas no son binarias.

Al igual que en la red de Hopfield, el aprendizaje no es realizado mediante sucesivas adaptaciones de sus pesos, sino que por una asignación dependiente de los patrones de ejemplo.

Teoría de la Resonancia Adaptiva Binaria (ART1)

La red ART1, introducida por Carpenter y Grossberg consiste en dos capas: Fl y F2, clasificadoras de cercanía, las que almacenan un número arbitrario de patrones espaciales binarios $A_k=\{a_1^k,a_2^k,...,a_n^k\}$, k = 1, 2,..., m, usando un aprendizaje competitivo en Línea.

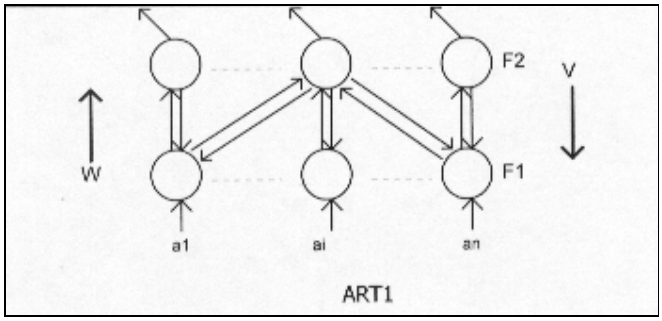


Figura 2.4.8

La capa F1 está constituida por un cantidad de neuronas igual al número de componentes del patrón de entrada y está totalmente interconectada con la capa F2 por medio de dos tipos de conexiones: las conexiones que conducen de F1 a F2 denominadas W_{ij}, y las conexiones que conducen de F2 a F1 denominadas V_{ij}. La función de umbral utilizada por las neuronas de F2 corresponde a una sigmoidal.

Como esta red opera en Línea, la descripción de su entrenamiento y aprendizaje quedará determinada por la operación de la red. El método

que se describirá a continuación corresponde al método rápido de aprendizaje y consiste, básicamente, en los siguientes pasos:

Paso 1. Inicialización de los pesos.

$W_{ij}(0) = 1 / (1+N)$

$V_{ij}(0) = 1 \quad ,0 \leq i \leq N\text{-}i, \quad 0 \leq 1 \leq M\text{-}1$

Asignar $\rho \quad ,0 \leq \rho \leq 1$

donde N es el número de entradas de la capa F1 y M es el número de elementos en la capa F2. W_{ij} son las conexiones de F1 a F2 y V_{ij} de F2 a F1, entre el nodo de entrada i y el nodo de salida j al tiempo 0. El valor de ρ es el umbral! de vigilancia que indica qué tan próxima una entrada debe ser para ser almacenada.

Paso 2. Aplicar una nueva entrada.

Se presenta un patrón de entrada $Ak=\{a_1^k,a_2^k,...,a_n^k\}$, k = 1, 2,..., m a F1, quedando su salida puesta con los valores de la entrada:

$F1 = \{ a_1^k,a_2^k,...,a_n^k\}$

Paso 3. Cómputo de las salidas de F2.

Cada elemento de proceso de F1 activado con un valor a_i envía dicha señal a través de las conexiones W_{ij} a F2.

Paso 4. Seleccionar el ganador de la competencia.

Cada elemento de proceso de F2 compite con los otros quedando solo un elemento activo, seleccionado según el criterio:

$$\mu_j^* = \max_j \{\mu_j\}$$

Paso 5. Propagación hacia atrás.

El elemento ganador en F2 propaga su señal hacia F1 por medio de las conexiones V_{ij}, generando un nuevo conjunto de activaciones en F1, al que llamaremos:

$$F1' = \{ a_1', a_2', \ldots, a_n' \}$$

donde $a_i' = \mu_j^* . V_{ij}$

Paso 6. Término del aprendizaje del patrón.

En este paso, el aprendizaje del patrón ha concluido y la red está lista para recibir al siguiente. Previo a esto, se debe habilitar todos los nodos que fueron deshabilitados en el paso anterior.

2.4.4. Redes con aprendizaje supervisado y evocación retroalimentada.

Estado cerebral en una caja (Brain-State-in-a-Box. BSB)

La red BSB fue introducida por Anderson, Silverstein, Ritz & Jones en 1977 y consiste, básicamente, en una red de una sola capa de neuronas, auto asociativa, clasificadora de cercanía. El número de neuronas queda determinado por la cantidad de componentes del vector de entrada y sus conexiones se distribuyen por todas las neuronas, inclusive a si misma.

A la salida de cada neurona existe una función de activación del tipo rampa con saturación, definida como:

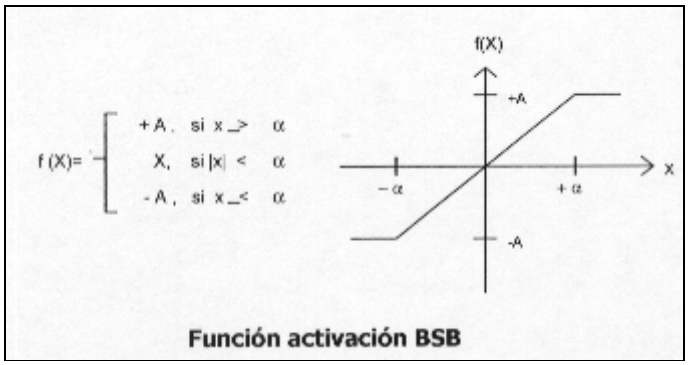


Figura 2.4.9

[Tornado Computer Based Training on Neural Networks, Jackes]

Esta red almacena patrones espaciales análogos arbitrarios, Ak= $\{a_1k,a_2k,...,a_nk\}$, usando un aprendizaje por corrección del error. Su aprendizaje es realizado fuera de lìnea mediante un entrenamiento supervisado. El entrenamiento de la red consiste en presentar un conjunto de pares de entrada y salida de modo que el algoritmo de aprendizaje ajuste los pesos de la red con el fin de minimizar el error.

Una vez que el error se mantiene dentro de niveles bajos, se puede pasar a la etapa de operación o evocación de la red, donde se presentan patrones a la entrada generando la red una salida que dependerá de la siguiente ecuación:

$$a_i\,(t+1) = f(a_i(t) + \beta \sum_{j=1}^{n} W_{ij}.a_j\,(t))$$

donde $a_i(t)$ y $a_j(t)$ son los valores de los elementos de proceso i y j respectivamente, al tiempo t. *f* corresponde a la función de activación ya mencionada. Este proceso se ejecuta en forma repetitiva hasta que los valores de aj se estabilicen o dejen de cambiar.

[Tornado Cornputer Based Training on Neural Networks, Jackes]

2.5. Aplicación de Redes Neuronales Artificiales

Las redes neuronales artificiales han sido aplicadas a un número en aumento de problemas en la vida real y de considerable complejidad, donde su mayor ventaja es en la solución de problemas que son bastante complejos para la tecnología actual, tratándose de problemas que no tienen una solución algorítmica cuya solución algorítmica es demasiado compleja para ser encontrada.

En general, debido a que son parecidas al cerebro humano, las redes neuronales son bien nombradas ya que son buenas para resolver problemas que el humano puede resolver pero las computadoras no. Estos problemas incluyen el reconocimiento de patrones y la predicción del tiempo. De cualquier forma, el humano tiene la capacidad para el reconocimiento de patrones, pero la capacidad de las redes neuronales no se ye afectada por la fatiga, condiciones de trabajo, estado emocional y compensaciones.

Un área en la que las redes neuronales artificiales han tenido éxito ha sido el reconocimiento óptico de caracteres (IOC) ya sea impresos o manuscrito. El proceso de reconocimiento óptico no es trivial e implica el uso de técnicas sofisticadas de varios tipos. Por ejemplo, se requiere de

técnicas de procesamiento de imágenes para convertir las escalas de gris en blanco y negro y representar estas imágenes con números binarios. Es muy probable que se requiera también de técnicas para traslapados. Muchas veces el proceso de obtención de las imágenes tiene problemas de alineación por lo que los caracteres deben ser rotados. Finalmente, habrá una etapa de reconocimiento de patrones de los datos binarios y es donde las redes neuronales se han usado más.

Dentro de las múltiples áreas de aplicación para las redes neuronales artificiales, podemos citar algunos ejemplos tales como:

Control de la eficiencia de una máquina

El comportamiento de la máquina de un carro es influenciada por un gran número de parámetros como temperatura, mezcla de combustible, viscosidad de lubricante, etc. Rolls Royce ha usado redes neuronales para ajustar dinámicamente una máquina de acuerdo a su estado actual.

Predicción de bolsa de valores

El triunfo o no en el uso real en la predicción de mercados es difícil de alcanzar desde que las compañías usando estas técnicas son

comprensiblemente reacias a mostrar información. Algunos creen que el uso de redes neuronales en este ambiente es un truco de mercadeo.

Reconocimiento de firmas

Cada persona tiene una firma distinta. Aunque cada vez que firmamos algo, la firma es Ligeramente distinta, la forma en general es similar y puede ser reconocida por un experto humano. Una compañía ha creado una máquina que reconoce firmas con un gran nivel de precisión. Toma en cuenta la velocidad en que se firma, además de la presión, grosor y otros factores.

Otras aplicaciones tales como reconocimiento óptico de caracteres, control de procesos industriales, aplicaciones de predicción del tiempo, decisiones sobre otorgamientos de préstamos, Análisis de inversiones, Análisis de firmas, Monitoreo y Mercadotecnia. Los sistemas neuronales han sido ampliamente aceptados en campos de aplicación tales como:

Finanzas.

- Predicción de índices
- Detección de fraudes.
- Riesgo crediticio, clasificación
- Predicción de la rentabilidad de acciones

Negocios

- Marketing
- Venta cruzada
- Campañas de venta

Tratamiento de textos y proceso de formas.

- Reconocimiento de caracteres impresos.
- Reconocimiento de gráficos.
- Reconocimiento de caracteres escritos a mano.
- Reconocimiento de escritura manual cursiva.

Alimentación

- Análisis de olor y aroma.
- Perfilamiento de clientes en función de la compra.
- Desarrollo de productos.
- Control de Calidad.

Energía

- Predicción de consumo eléctrico
- Distribución recursos hidráulicos para la producción eléctrica
- Predicción de consumo de gas en la ciudad

Industria manufacturera.

- Control de procesos.
- Control de calidad.
- Control de robots.

Medicina y salud

- Ayuda al diagnostico.
- Análisis de Imágenes.
- Desarrollo de medicamentos.
- Distribución de recursos.

Ciencia e Ingeniería

- Análisis de datos y clasificación
- Ingeniería Química.
- Ingeniería Eléctrica.
- Climatología.

Transportes y Comunicaciones.

- Optimización de rutas.
- Optimización en la distribución de recursos

CAPÍTULO 3

3. MODELO DE KOHONEN

3.1 REDES KOHONEN

La red neuronal de Kohonen es una técnica ordenada de asociación que permite proyectar puntos multidimensionales a una red de dos dimensiones. Hay dos conceptos dominantes importantes en la comprensión de un modelo de Kohonen; son el aprendizaje competitivo y la autoorganización.

El aprendizaje competitivo es encontrar simplemente una neurona que sea la más similar al modelo de la entrada de información. La capacidad de autoorganización implica que la red tiene la facultad de modificar a la nueva neurona y a la vecindad de la neurona tal que sea aún más similar a ella.

Se ha demostrado que nuestro cerebro organiza nuestras neuronas en zonas, de tal forma que las informaciones captadas del entonces a través de los órganos sensoriales (sentidos) se representan internamente en forma de capas bidimensionales. Por ejemplo, en el sistema visual se

han detectado mapas del espacio visual en zonas de cortex (capa externa del cerebro).

[Tornado Self Organization and Assoclative Memory, Kohonen]

Las evidencias sugieren que esta agrupación u organización neuronal está predeterminada genéticamente, pero es muy probable que este tipo de funcionalidad se origine mediante el aprendizaje. Esto sugiere, por tanto, que el cerebro podría poseer la capacidad inherente de formar mapas topológicos de las informaciones recibidas de exterior.

De hecho, esta teoría podría explicar su poder de operar con elementos semánticos: algunas áreas del cerebro simplemente podrían crear y ordenar neuronas especializadas o grupos con características de alto nivel y sus combinaciones. Se tratarla, en definitiva, de construir mapas espaciales para atributos y características.

[Tornado Kohonen Maps, Kaski]

Teuvo Kohonen presentó en 1982, basándose en estas ideas, un sistema que incorpora este comportamiento de agrupación y organización. Se trataba de un modelo de red neuronal con capacidad para formar mapas de características de manera similar a como ocurre en el cerebro.

Kohonen trata de demostrar con este tipo de sistema o red que una entrada externa por si sola, suponiendo una estructura propia y una descripción funcional del comportamiento de la red, era suficiente para forzar la formación de mapas.

El modelo de Kohonen se presenta en 2 variantes, denominadas LVQ (Learning Vector Quantization) y TPM (Topology-Preserving Map) o SOM (Self-Organizating Map). Ambos modelos se basan en el principio de formación de mapas topológicos para establecer características comunes entre las informaciones (vectores) de entrada a la red, aunque difieren en las dimensiones de éstos, siendo de una sola dimensión en el caso de LVQ, y bidimensional, e incluso tridimensional, en la red TPM.

En cuanto al modelo Learning Vector Quantization LVQ, se presenta una red con un cierto número de neurona de entrada y salida. Cada una de las neuronas de entrada se conecta a las de salida a través de conexiones hacia adelante (también llamadas feedfoward).

El modelo de Mapas topológicos SOM trata de establecer una correspondencia entre los datos de entrada y un espacio bidimensional de salida, creando mapas topológicos de dos dimensiones, de tal forma que

ante datos de entrada con características comunes se deben activar neuronas situadas en zonas próximas de la capa de salida.

Este estudio se enfoca al segundo modelo sobre las redes neuronales basadas en Mapas de características Autoasociativo (SOFM).

[Tornado Self Oran,zation andAssoc,ative Mernorn Kohonen]

Teuvo Kohonen es uno de los investigadores más famosos y más prolíficos de neurocomputing, y él ha inventado una variedad de redes. Pero mucha gente se refiere a las “redes de Kohonen” sin especificar qué clase de red de Kohonen, y esta carencia de precisión pueden conducir a la confusión. La frase “red de Kohonen” refiere lo más a menudo posible a una red del tipo:

SOM: Self Organizing Map, redes competitivas que proveen una asociación topológica desde el espacio de entrada de información a un grupo de salida conocido como racimos (agrupaciones).

En un SOFM, las neuronas (racimos) se ordenan en una rejilla o plano, generalmente de dos dimensiones, pero a veces unidimensional o (raramente) tres o n-dimensional. La rejilla existe en un espacio que esté a parte del espacio de la entrada de información; cualquier número de

entradas de información se puede utilizar mientras el número de entradas de información es mayor que la dimensionalidad del espacio de la rejilla.

En un modelo SOFM, el fundamento básico es incorporar a una regla de aprendizaje competitivo un cierto grado de sensibilidad con respecto al vecindario o entorno. Esto hace que el número de neuronas que no aprenden desaparezca y ayuda a que se destaquen propiedades topológicas que aparezcan en el "mapeado" de características.

Asumamos que tenemos un modelo con las siguientes características:

- Un vector de entrada con N características, representado por un vector X
- Un espacio de patrones N-dimensional.

[Tornado Self Organization and Assoclative Memory Kohonen]

La red mapea el patrón de entrada hacia un espacio de salida. Por ejemplo, el espacio de salida puede ser un arreglo unidimensional o bidimensional de nodos de salida, que posee cierto orden topológico. La tarea es entrenar el modelo para que esa relación de orden se mantenga. Kohonen propuso que las neuronas de salida interactuaran entre Si lateralmente, llegando así a los mapas de características autorganizativos.

La característica principal en un modelo de este tipo es el aprendizaje de las neuronas, las cuales deben entrenarse para residir en un vecindario de neuronas ganadoras. El tamaño de este vecindario decrece en función del número de iteraciones.

Las neuronas del modelo deben tener una etapa o fase de entrenamiento para lograr formar este vecindario. Esta etapa de entrenamiento tiene dos pasos fundamentales:

Primer paso: Se debe seleccionar la neurona ganadora, es decir, la neurona más cercana al vecindario, midiendo la distancia Euclidea entre los vectores de pesos y el vector de entrada.

Segundo paso: Sea i^* el indicador del ganador, y sea I^* el conjunto de indicadores correspondiente a un vecindario definido del ganador i^*. Entonces los pesos asociados con el ganador y su vecindario se adaptan de la siguiente forma:

$$\Delta w_j = \eta(x - w_j)$$

para todos los indicadores j que pertenecen al conjunto I^*, y donde η es una pequeña constante positiva. La cuantía de la adaptación se puede escalar de acuerdo a una "función de vecindad" preestablecida, *v(j,i*).*

3.2. RECONOCIMIENTO DE PATRONES

El reconocimiento de patrones es un concepto muy general con un número grande de aplicaciones en ciencia y tecnología: lectura automática de caracteres escritos o impresos, reconocimiento automático de palabras habladas y de caras, identificación de partículas elementales en física de energía, etc.

Una de las principales aplicación de las redes neuronales de Kohonen es el reconocimiento de patrones.

3.3. MODELO “Self Organizing Map”

Durante la fase del entrenamiento, los vectores de los pesos son actualizados de manera que tiendan a acercarse al centro del racimo de los datos de entrada de información. Después de entrenarse, SOM produce un mapa topológico de racimos.

El mapa topológico tiene la característica que las neuronas cercanas de la salida del mapa representan racimos similares. Este mapa se puede sucesivamente utilizar para agrupar en una clase dada todos los racimos que tienen características similares.

En el gráfico 3.3.1 se muestra esquemáticamente como el modelo organiza los racimos y la técnica matemática para lograr esta asociación.

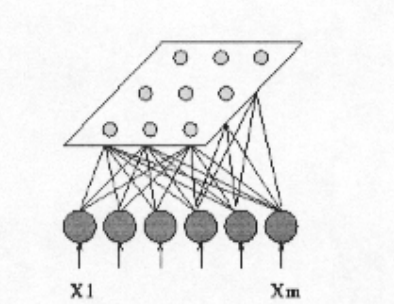


Figura 3.3.1

Donde:

- La capa de entrada se denomina Capa X, La capa de salida Capa Y, y los pesos Wi para cada i-entrada.
- Las distancias entre señales de entrada son calculadas:

  $Y_i = \|X - Y_I\|$

- La señal ganadora es $Y_k = min Y_i$

Hay muchos métodos que se pueden utilizar para reducir la dimensionalidad de los datos. Los datos se asocian a un espacio dimensional, típicamente a una o dos dimensiones. En la agrupación de racimos los vectores de datos similares se asocian a las neuronas próximas y por lo tanto la topología de los datos originales se preserva. Estas características son muy útiles en análisis de datos.

El número de neuronas en un SOM necesita ser bastante grande de modo que haya algunas neuronas para representar cada grupo de datos.

Estos conjuntos pequeños de neuronas proporcionan una representación simbólica a los grupos de los datos. En el proceso del entrenamiento estos conjuntos pequeños de neuronas se convierten en los prototipos.

3.4. ALGORITMO DE KOHONEN

El algoritmo self-Organizing de Kohonen, también llamado Mapas de Kohonen, es uno de los algoritmos artificiales mas conocidos de las redes neuronales.

En contraste con la mayoría de los otros algoritmos, se basa en aprendizaje y entrenamiento no supervisado.

Los mapas de Kohonen son una clase única de redes neuronales, puesto que construyen los mapeos topológicos de los datos del entrenamiento donde la localización de una unidad lleva la información semántica.

Por lo tanto, la aplicación principal de este algoritmo es el agrupar los datos, obteniendo una visualización de dos dimensiones del espacio de la entrada de información.

Los mapas self-Organizing contienen dos capas:

Una capa unidimensional que corresponde a la entrada de información y una capa competitiva de dos dimensiones, ordenada como una rejilla de unidades de 2 dimensiones.

Cada unidad de la capa mantiene un vector del peso (referencia), que, después de entrenar, se asemeja a un diferente modelo de la entrada de información.

[Tornado Self Organization and Associative Memory. Kohonen]

El algoritmo de aprendizaje de SOM logra dos objetivos importantes:

1. Agrupar los datos de entrada.
2. Ordenar espacialmente los datos de entrada del mapa de modo que los modelos similares a la entrada de información tiendan a producir

una respuesta en las unidades que están cerca una de otra en la rejilla.

Antes de comenzar el proceso de aprendizaje, es importante inicializar la capa competitiva con vectores normalizados. Los vectores del modelo de entrada de información se presentan a todas las unidades competitivas en paralelo y la mejor unidad (más cercana) se elige como el ganador.

Para cada vector la siguiente secuencia de pasos ocurre:

- Encuentre el nodo k al que el vector de peso está más cerca al vector actual de la entrada de información, basándose en la distancia euclidea.
- Entrene al nodo k y todos los nodos en alguna vecindad de k.
- Después de cada M ciclos, disminuye el tamaño de la vecindad.

CAPÍTULO No. 4

4. HERRAMIENTAS DE SOFTWARE

4.1 HERRAMIENTAS UTILIZADAS

Dado que La problemática del factor P involucra 5 variables como entradas a La red neuronal, se ha decidido desarrollar un paquete de software que permita visualizar una red de Kohonen con 5 entradas y convertirlas a un sistema de salida con una única variable.

La principal ventaja de desarrollar es la opción de personalizar y hacer un sistema a la medida que permita visualizar lo mejor posible los resultados de la red neuronal partiendo de un diseño basado en el problema planteado.

El factor limitante en este desarrollo, el cual también lo encontramos en las aplicaciones existentes en el mercado, es que los mapas topográficos se muestran en planos de 2 o 3 variables. En nuestro caso, nuestros mapas topográficos contienen 5 variables. Debido a esto se hace difícil esquematizar un escenario que grafique el sistema con estos números de variables. Para tratar de llevar un control sobre los

mapas se implementa en el sistema un registro que almacena todas las variables con sus variaciones durante la ejecución de la red.

4.2 ANÁLISIS DE HERRAMIENTAS

Uno de los objetivos principales de esta tesis es conocer las aplicaciones existentes en el mercado para la solución de sistemas neuronales.

Hemos hecho un estudio de las principales herramientas existentes, dividiéndolas en paquetes de software que son de libre uso y paquetes de software con un valor comercial. A continuación presentamos las herramientas más utilizadas de acuerdo a la clasificación planteada.

Entre las más conocidas, destacando en su interfase con el usuario, tenemos 2 herramientas muy utilizadas para crear un sistema neuronal: NEUROSOLUTIONS y SPSS.

NEUROSOLUTIONS

NeuroSolutions se basa en el concepto que las redes neuronales se pueden analizar en un conjunto fundamental de componentes. Estos componentes son individualmente simplistas, pero varios componentes

conectados juntos crean un sistema y pueden dar lugar a las redes capaces de solucionar problemas muy complejos.

La interfaz del usuario en NeuroSolutions fue inspirada por el proceso de diseñar un circuito electrónico. Los componentes electrónicos, tales como resistores, condensadores, y los transistores, se presentan en un tablero y se interconectan con alambre; juntos para formar un circuito. El circuito es probado con señales de entrada y se esperan respuestas.

NeuroSoiutions proporciona todas las herramientas para construir una red neuronal utilizando todos los componentes, tales como axiomas, sinapsis, etc. Los componentes de entrada de información se utilizan para ingresar señales, y los componentes de salida se utilizan para visualizar la respuesta de la red.

La mayoría de los productos de software de redes neuronales ofrecen simplemente un simulador, el cual da un número pequeño de modelos y un conjunto pequeño de parámetros con los cuales configurarlos. En este caso se tienen un control total sobre La parametrización de los componentes que conforman la red neuronal.

La pantalla para la creación de los modelos basándose en componentes se ilustra en la siguiente figura:

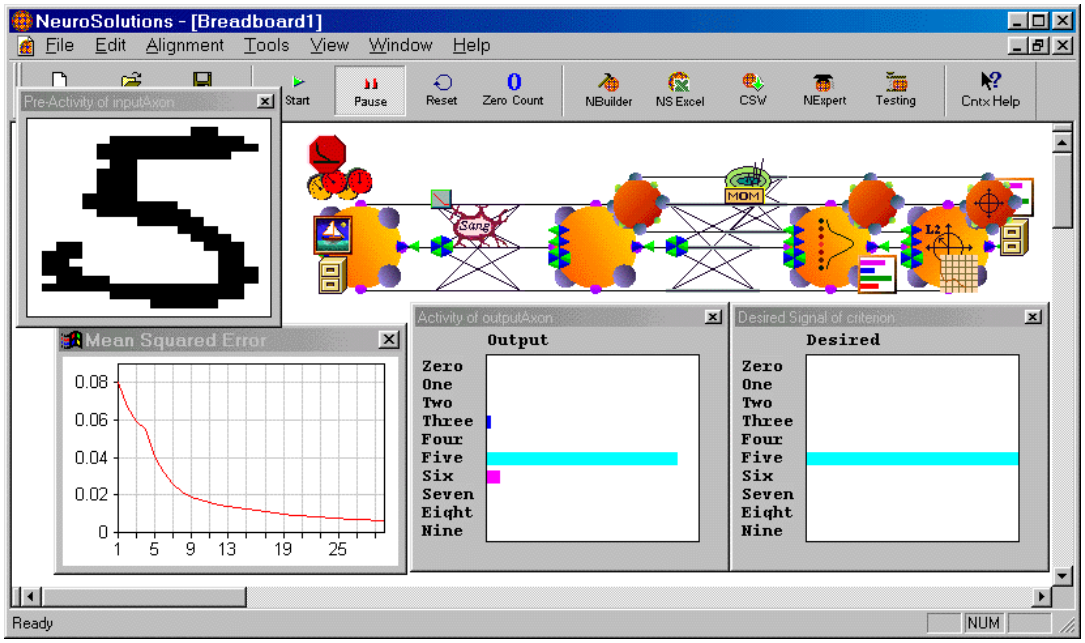


Figura 4.1.1

Este ejemplo es un problema del reconocimiento de caracteres óptico (OCR) que se soluciona utilizando una red neuronal, donde la entrada de información a la red es un conjunto de imágenes 24x18 de dígitos manuscritos. Cada imagen tiene una salida deseada correspondiente, que es una codificación del digito que la imagen representa.

SPSS

El software SPSS contempla una solución completa de redes neuronales y análisis estadístico; lee y escribe directamente archivos de SPSS, por lo que podrá realizar análisis detallados de los resultados y explorar los modelos con mayor detalle. Permite crear los modelos de redes neuronales más utilizados como son el Perceptron, mapas topográficos, etc. SPSS dispone de una útil ayuda que le permite empezar a trabajar en el menor tiempo posible. Utilice la completa ayuda en pantalla y dos manuales de Neural Connection: el manual del usuario, con un tutorial, operaciones y algoritmos; y una guía de aplicaciones de las redes neuronales.

Su interfase se presenta como en la siguiente gráfica:

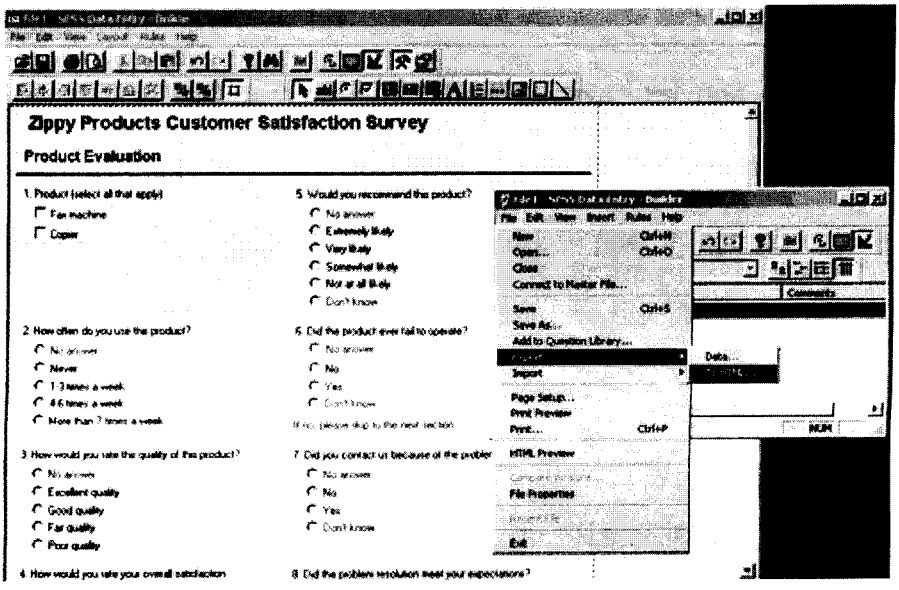


Figura 4.1.2

Software de licenciamiento gratituo

Dentro de las principales herramientas de libre uso, es decir, de licenciamiento gratituo, encontramos las siguientes:

1. NeurDS
2. GENESIS
3. DartNet
4. SNNS
5. Aspirin/MIGRALNES
6. ALN Workbench
7. PDP++
8. Neural Networks at your Fingertips
9. Nenet v1.0

NeurDS

Neural Design and Simulation System. Es una herramienta de propósitos generales para construir, ejecutar y analizar modelos de redes neuronales de manera eficiente. NeurDS puede compilar y ejecutar cualquier Modelo Neuronal utilizando una interfase bastante consistente y amigable o utilizando una interfase orientado a comandos. Su código fuente se encuentra disponible en:

http://*hpux*.u-alzu.ac.jp/hppd/hpux/NeuralNets/NeurDS-3.1/

http://askdonna.ask.unikarlsruhe.de/hppd/hpux/NeuralNets/NeurDS-3.1/

GENESIS

GENESIS 2.0 (General Neural Simulation System) es una plataforma de simulación La cual ha sido desarrollada para soportar La simulación de sistemas neuronales para modelos desde simples neuronas hasta modelos de alta complejidad. Se ejecuta sobre plataformas Unix y utiliza como interfase gráfica Xodus.

Disponible desde ftp://genesis.bbb.caltech.edu/pub/genesis. Más información sobre Genesis se puede encontrar en el sitio http://www.bbb.caltech.edu/GENESIS/

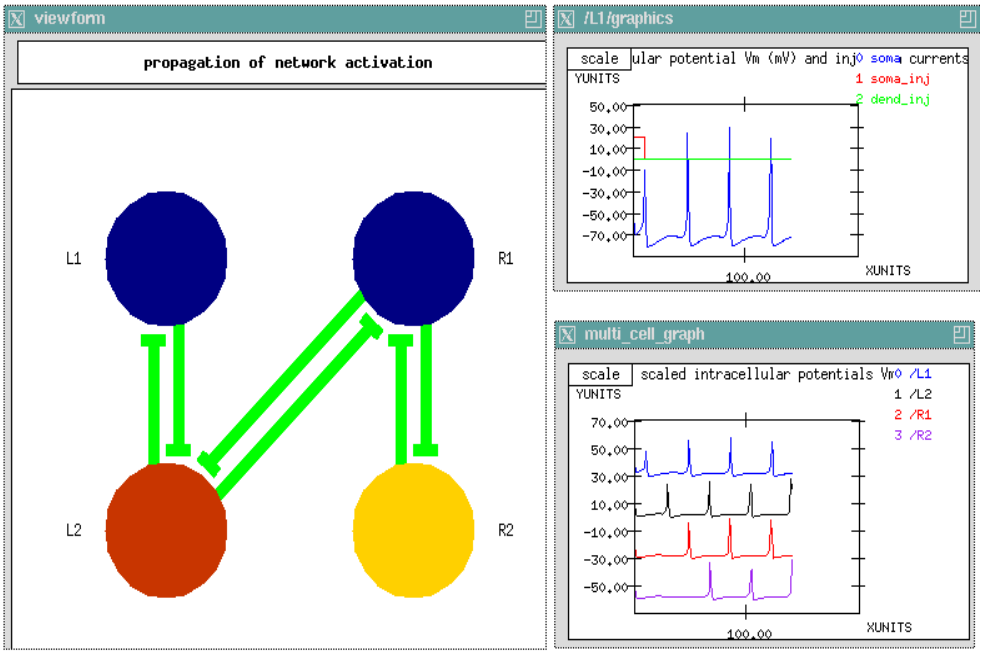


Figura 4.1.3

DartNet

DartNet es un simulador basado en el modelo de Backpropagation desarrollado para una plataforma Macintosh por Jamshed Bharucha y Sean Nolan. Esta herramienta presenta una interfase gráfica para Mac y provee un número de utilitarios para construir, editar, entrenar y probar una red neuronal basada en propagación hacia atrás.

Este programa está disponible para descargar a través de un ftp anónimo en La dirección ftp.dartmouth.edu ftp://pub/mac/dartnet.sit.hgx/

Aspirin/MIGRALNES

Aspirin/MIGRALNES 6.0 consiste en un generador de código que construye simulaciones de redes neuronales basándose en la descripción escrita en un lenguaje llamado Aspirin. Una interfase llamado “Migraines” es proporcionada para exportar datos desde La red neuronal a las herramientas de visualización.

Los usuarios pueden visualizar los datos usando las herramientas públicas o comerciales de gráficas y análisis para plataformas Unix. Este software está disponible desde el sitio ftp.cognet.ucla.edu [128.97.50.19] en /pub/alexis/am6.tar.Z.

SNNS 4.1

Stuttgarter Neural Network Simulator creado por La Universidad de Tuebingen, posee una interfase que visualiza las redes en 2 y 3 dimensiones. Actualmente soporta modelos de Backpropagation, Quickprop, Backpercolation, Funciones Radiales (RBF), ART1, Correlación, LVQ, Redes Hopfield, Redes Jordan y Elman, Memoria Autoasoclativa, Kohonen — Self organizing maps. Todos los modelos auto generan su código en C. Operan sobre las plataformas SunOS, Solaris, IRIX, Ultrix, OSF, ALX, HP/UX, NextStep, Linux, y Windows 95/NT. Mayor información se puede encontrar en el sitio http://www-ra.informatik.uni-tuebingen.de/SNNS/

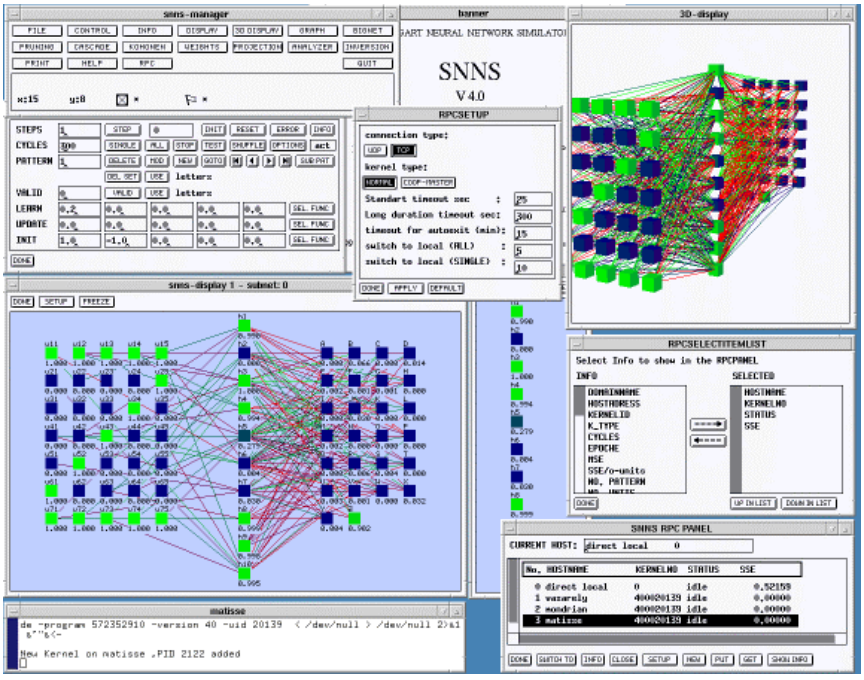


Figura 4.1.4

ALN Workbench

ALNBench es un programa de hoja de cálculo para MS-Windows (NT, 95) que permite al usuario importar el entrenamiento y pruebas para predecir una columna de datos basándose en el entrenamiento utilizado. Es un programa fácil de utilizar para la investigación, la educación y la evaluación de la tecnología ALN. Cualquier persona que puede utilizar una hoja de cálculo puede entender rápidamente cómo utilizarla.

Un ALN consiste en funciones lineales con pesos adaptables en las hojas de un árbol de operadores máximos y mínimos. El árbol crece automáticamente durante el entrenamiento. Este programa puede se descargado desde el sitio http://www.dendronic.com/beta.htm

PDP++

El software PDP++ es un nuevo sistema para la simulación de redes neuronales escrito en C++. Es bastante fácil para los usuarios principiantes, con gran alcance y flexible para el uso de la investigación. La versión actual es 1.0. Se ha probado sobre Unix con X-Windows. Entre sus principales características tenemos la interfase gráfica InterViews, visores de tiempo real, visores de datos, diseño orientado a objetos.

Utiliza los algoritmos Feedforward y máquina de Boltzmann, Hopfield. Este software puede ser descargado vía un ftp anónimo en ftp://cnbc.cmu.edu/pub/pdp++/

Neural Networks at your Fingertips

Neural Networks at your Fingertips es un paquete aplicativo que contiene el código fuente para simulación de redes neuronales. Este paquete consiste de ocho programas, cada uno de los cuales implementan una arquitectura en particular. Soporta a arquitecturas tales como redes Adeline, Backpropagation, Hopfield, Máquina de Boltzmann, Kohonen Self-Organizing Map, Redes tipo Adaptive Resonance Theory. Este paquete de software se encuentra disponible en el sitio www.geocities.com en La dirección http://www.geocities.com/CapeCanaveral/1624

Nenet v1.0

Nenet v1.0 es una aplicación basada en Windows diseñada para facilitar el uso del algoritmo de mapas auto organizados “Self-Organizing Map (SOM)”. Implementa el algoritmo SOM y posee una interfase que permite visualizar los mapas desde 5 diferentes métodos. Los parámetros de las neuronas pueden ser fácilmente editados.

Este software se encuentra disponible en La dirección http://www.mbnet.fi/~phodju/nenet/nenet.html.

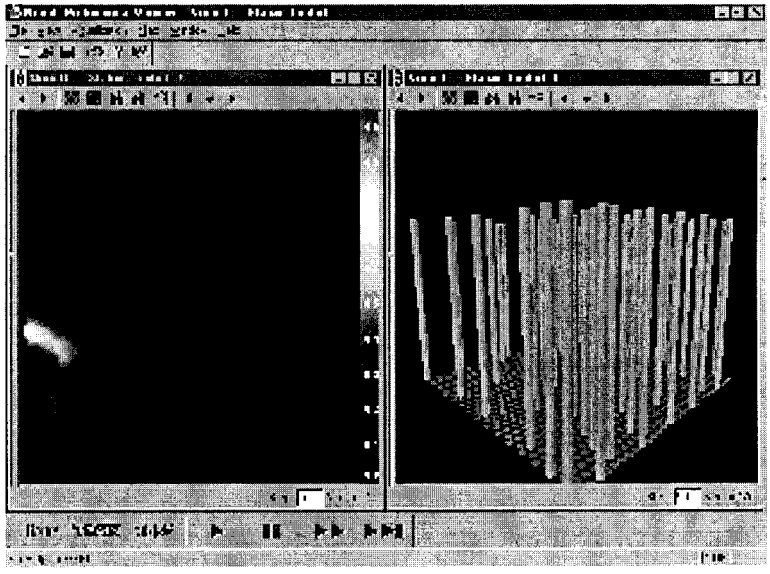


Figura 4.1.6

Software Comercial

Dentro de las principales herramientas comerciales, encontramos las siguientes:

1. NeuralWorks
2. NeuroForecaster
3. Ward Systems Group (NeuroShell, etc.)
4. NeuroSolutions v3.0
5. hav.Software: havBpNet++, havFmNet++, havBpNet:J

6. NeuroGenetic Optimizer (NGO) Version 2.0
7. Neural Connection
8. Pattern Recognition Workbench Expo/PRO/PRO+
9. Trajan 2.1 Neural Network Simulator
10. Viscovery SOMine
11. Alnet

NeuralWorks

NeuralWorks es una herramienta que permite crear soluciones de redes neuronales para modelos Backpropagation, Art, Kohonen, Regresion, Fuzzy Art-map, Redes probabilísticas, Kohonen self-organizing map, Redes Lvq, Máquina de Boltmann, etc...

Sistema Operativo: PC, Sun, IBM rs6000, Apple Macintosh, SGI, Dec, HP.

Requerimientos: PC: 2MB de memoria y 6MB de espacio físico.

Precio Aproximado: Depende de La plataforma (US$ 9.150- 14.500)

Información Adicional: URL: http://www.neuralware.com/

NeuroForecaster & Visua Data

NeuroForecaster es una herramienta de Redes neuronales basada en ambiente MS-Windows diseñada para construir sofisticado pronósticos, tal como pronosticar el tiempo o indicadores. Entre sus principales características: Opera sobre 12 modelos Neuro — Fuzzy, es una herramienta multitarea, Maneja ilimitado número de redes y conexiones, Monitorea progreso de aprendizajes, guarda históricos de pesos, provee análisis de error.

Sistema Operativo: PC Ms-Windows 3.1, DOS 5.0

Requerimientos: PC: 4MB de memoria y 4MB de espacio físico.

Precio Aproximado: US$ 1.199 por usuario

Información Adicional:

http://www.singaporegateway.com/products/nfga/nf.htm

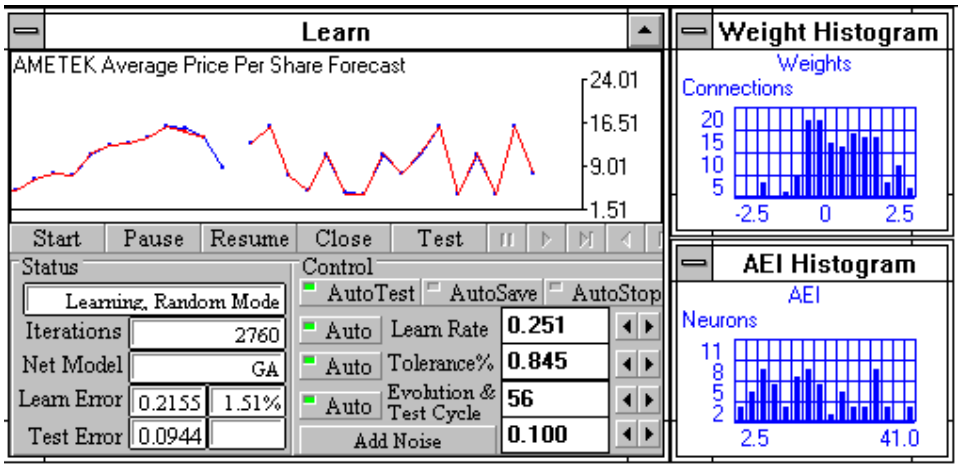


Figura 4.1.7

Ward Systems Group (NeuroShell. etc.)

NeuroShell es un grupo de utilitarios que cumple una específica función dentro de una solución neuronal. ***NeuroShellPredictor*** *es* utilizado para pronosticar y estimar cantidades numéricas tales como ventas, precios, carga de trabajo, nivel, costos, cuentas, velocidad, capacidad, etc. Lee y escribe ficheros de texto.

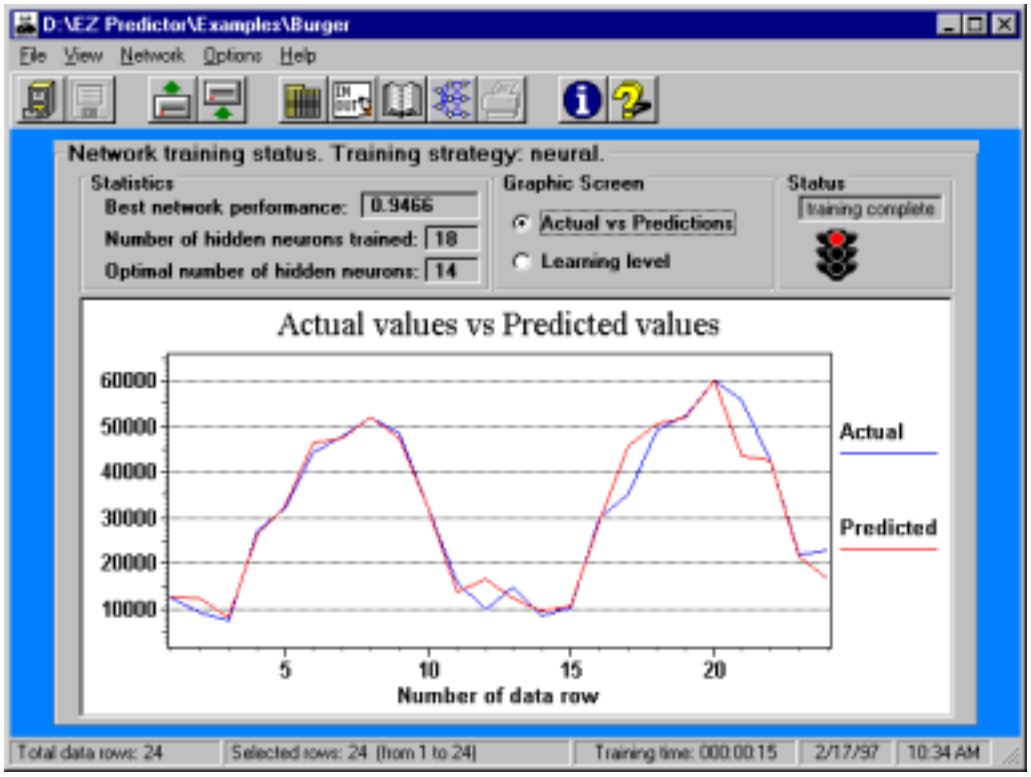


Figura 4.1.8

NeuroShell Classifier soluciona los problemas de clasificación y categorización basados en patrones aprendidos de datos históricos. ***Neuro Shell Run Time*** *Server* permite distribuir redes creadas con NeuroShell Predictor o NeuroShell Classifier desde un interfase sencillo o

través de La hoja de cálculo de Excel. ***NeuroShell*** *GeneHunter* es una algoritmo diseñado para optimizaciones tales como encontrar los mejores horario, los indicadores financieros, las variables modelo, etc.

NeuroShell Engine es un control Active X que contiene los métodos de aprendizaje para La red neuronal que han sido usados en NeuroShell Predictor y Classifier. Estos métodos están disponibles para ser integrados con sus propias aplicaciones por cualquier programador.

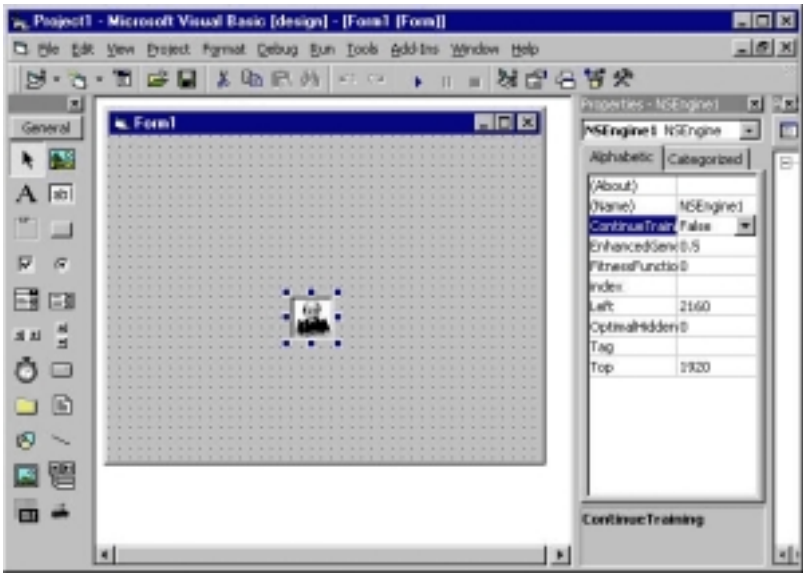


Figura 4.1.9

Dentro de sus principales aplicaciones tenemos:

Negocios y Aplicaciones Financieras: predicción del mercado, pronósticos del precio, pronósticos de las ventas, estrategias legales, proceso de las solicitudes de crédito, etc.

Aplicaciones Científicas: Exploración petrolífera, órdenes de la materia prima, identificación de polímeros, diagnósticos de problema en circuitos, identificación de productos, control de calidad, identificación química, etc.

Aplicaciones Médicas: diagnosis de enfermedades, análisis resultados, diagnosis psiquiátrica, análisis de pruebas médicas, etc.

Sisterna Operativo: PC Ms-Windows 3.1, Windows 95 & 98, Windows NT

Requerimientos: PC: 16 MB de memoria y 25MB de espacio físico

Precio Aproximado: US$ 395 por utilitario

Información Adicional: URL: http //www.wardsystems.com

NeuroGenetic Optimizer (NGO) Version 2.0

NeuroGenetic Optimizer es una herramienta de desarrollo de redes neuronales que usa algoritmos genéticos para optimizar las entradas y estructuras de La red. NGO busca la más óptima solución para una red. NGO esta equipada para predecir información del tiempo, ventas, costos, procesos, etc. Posee una fácil interfase gráfica para Windows, de tal manera que los algoritmos de modelamiento y búsquedas en la red son muy fáciles de manipular.

Sicterna Operativo: PC Win 3.1, Windows 95, Windows 98, Windows NT

Requerimientos: PC: 8 MB de memoria y 15MB de espacio físico

Precio Aproximado: Desde US$195

Información Adicional: URL: http://www.bio-comp.com.

NeuroSolutions v3.0

NeuroSolutions es un simulador altamente gráfico de redes neuronales para ambientes basados en Windows 95/98 y Windows NT 4.0. Este software combina una interfase modular con los procedimientos avanzados de aprendizaje, tales como backpropagation. El resultado es un ambiente para diseñar redes neuronales para la investigación o solución de problemas reales.

Este software se basa en el manejo de archivos DLL que pueden ser accesados tanto por la aplicación como por aplicativos como Excel, Access, Visual Basic, de tal forma que tiene una total integración con aplicaciones de tipo OLE. Adicionalmente, posee un generador de código en C++.

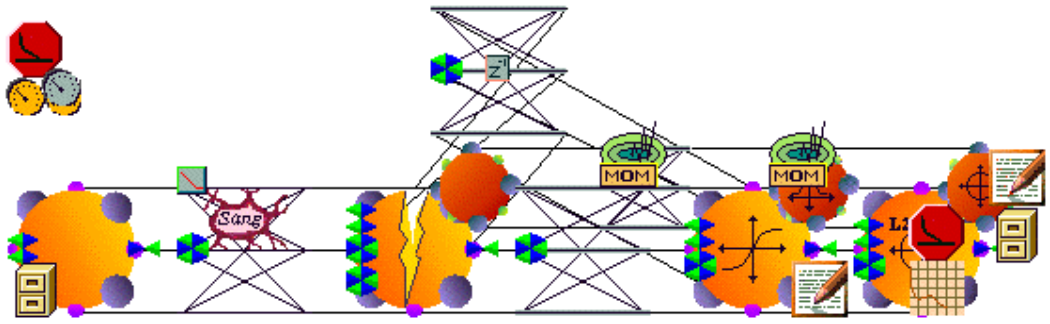


Figura 4.1.10

Este aplicativos maneja una serie de topologías tales como: Perceptron Multicapa, Redes Feedforward, Redes de tipo Jordan-Elman, Redes basadas en Mapas Organizados (SOFM), Redes de tipo Función Radial (RBF), e inclusive redes con topología definida por el usuario.

Sisterna Operativo: PC Windows 95, Windows 98, Windows NT 4.0

Requerimientos: PC: 16 MB de memoria y 25MB de espacio físico.

Precio Aproximado: US$ 1950

Información Adicional: URL: http://www.nd.com

HAV.Software: havBpNet++, havFmNet++

HavBpNet++ es una Clase (librería) en C++ que implementa el modelo de FeedForwarad a través de Backpropation. Soporta todos los estándares tales como pesos, momentos, banda de aprendizajes, etc. Además incluye 5 tipos de funciones de activación: lineal, Signoidal, Hiperbólica, Tangencial, Logarítmica.

Sistema Operativo: PC DOS, WIN 3.1, NT, Un/k, Hpux, Sun, AISS, Iris

Precio Aproximado: US$50 por desarrollador

Información Adicional: URL: http://www.hav.com/

HavFmNet++ es una librería en C++ que implementa redes de tipo Kohonen utilizando Mapas Auto-organizativos. Los mapas que se pueden crear con esta librería pueden tener desde 1 dimensión hasta la que se desee.

Sistema Operativo: PC DOS, WIN 3.1, NT, Un/k, Hpux, Sun, AISS, Iris

Precio Aproximado: US$ 50 por desarrollador / $*1000.00 Licencia corporativa*

Información Adicional: URL: http://www.hav.com/

Neural Connection

Neural Connection es una herramienta gráfica de redes neuronales en La cual los usuarios pueden crear un espacio de trabajo basado en iconos para construir modelos que permitan la predicción y clasificación de datos. Incluye utilitarios que permite el modelamiento y pronósticos de datos basándose en los modelos más conocidos de redes neuronales. Incluye un utilitario para crear redes neuronales tipo Multicapa Perceptron, RBF y Kohonen.

Adicionalmente posee 3 herramientas de análisis de datos. Los resultados de las redes generados en esta herramienta pueden ser

analizados a través de una interfase completamente gráfica o en archivos tipo texto.

Con esta herramienta se han desarrollado soluciones para predicción de datos en mercados, segmentación de mercados, predicción de la china, pronósticos de demandas y ventas de productos.

S/sterna Operativo: PC Win 3.1, Windows 95, Windows 98

Requerimientos: PC: 8 MB de memoria y 4 MB de espacio físico.

Precio Aproximado: US$ 995

Información Adicional: URL: http://www.spss.com

Pattern Recognition Workbench *Expo/PRO*/PRO+

Pattern Recognition Workbench (PRW) es una herramienta para resolver problemas basados en reconocimiento de patrones usando redes neuronales. Con una interfase flexible, intuitiva y gráfica, puede soportar arquitecturas y métodos de aprendizaje que incluyen Backpropagation, RBF, Kohonen. Adicionalmente posee un generador de código en C, lo cual genera DLLs, que permiten ligar las redes creadas con aplicaciones tales como Excel vía vínculos OLE. Los resultados de las redes pueden ser vistos a través de histogramas, grafos en 2 y 3 dimensiones.

Sisterna Operativo: PC Win3. 1, WFW3. 11, Windows 95/98 y Windows NT

Requerimientos: PC: 8 MB de memoria y 5 MB de espacio físico.

Trajan 2.0 Neural Network Simulator

Trajan 2.1 Professional es una red neuronal basada en Windows que incluye soporte a varios tipos de redes y algoritmos de aprendizajes. Está disponible en versión de 32 bits. Dentro de sus arquitecturas soportadas, incluye soporte a redes Multicapa Perceptrons, Redes Kohonen, Redes RBF. Incluyen algoritmos de aprendizaje tales como backpropagation y mapas auto-organizativos.

Utiliza una interfase sencilla basada en gráficos, barras y cálculos. Toda la información que maneja Trajan puede ser transferida a otras aplicaciones en Windows tales como hojas de cálculo. Adicionalmente este unitario posee una serie de archivos DLL que permite crear aplicaciones personales desde C o Visual Basic para crear y editar redes neuronales.

Sistema Operativo: PC Windows 95, Wndows 98 y Windows NT 4.0

Requerimientos: PC.. 8 MB de memoria y 15 MB de espacio físico.

Precio Aproximado: US$ 995

Información Adicional: URL: http://www.trajan-software.demon.co.uk

Viscoverv SOMine

Viscovery SOMine es una herramienta que permite la exploración de datos utilizando mapas autoorganizados implementados en un modelo de red neuronal de Kohonen. El mapa resultante de la red puede ser utilizada para identificar y evaluar características escondidas de datos. Los resultados son presentados en una interface gráfica lo cual le permite al usuario analizar los datos.

Sistema Operativo: PC Wüidows 95, Windows 98y Windows NT 4.0

Requerimientos: PC: 8 MB de memoria y 10 MB de espacio físico.

Precio Aproximado: US$ 1.495

Información Adicional: URL: http://www.eudaptics.co.at/

AINET

Ainet es una aplicación de redes neuronales que esta diseñada específicamente para facilitar las tareas de modelamiento de una red. Este aplicativo maneja una serie de topologías tales como: Perceptron Multicapa, Feedforward, Jordan-Elman, Mapas Organizados (SOFM), Función Radial (RBF), incluyendo redes definida por el usuario.

Sisterna Operativo: PC Windows 95, Windows 98 y Windows NT 4.0

Requerimientos. PC: 16 MB de memoria y 15MB de espacio físico.

Precio Aproximado: US$ 199

Información Adicional: URL: http://www.ainet-sp.si/ainetNN.htm

4.3 COMPARATIVO DE HERRAMIENTAS

La principal diferencia que podemos notar entre las herramientas presentadas en este trabajo es la orientación aplicativa claramente identificada en las aplicaciones de carácter comercial. La mayoral de soluciones gratuitas buscan un fin educativo y permite manipular cualquier sistema dejando abierta las opciones para crear cualquier tipo de red neuronal sin importar el uso práctico que se le quiera dar.

A diferencia, las herramientas comerciales todas contemplan un campo aplicativo en el mercado, tales como, investigación científica, mercado, predicción de clima, etc.

Por motivos netamente investigativos esta tesis proporciona una herramienta desarrollada en Visual Basic, orientada a objetos y métodos, de tal forma que se programaron los procesos y componentes de la red neuronal. En cuanto a la parte técnica de las herramientas, se nota un claro desarrollo sobre plataformas Unix, la cual permite manejar un mejor desempeño para cantidades considerables de datos. A continuación se muestra una tabulación de las herramientas mostradas y sus características y precios, con el propósito de identificar claramente las potencialidades y accesibilidad de los productos.

Herramienta	Interface Gráfica	Métodos	Plataforma	Código	Valor	Aplicación
NeuroDS	NO	>10	UNIX	NO	Gratis	Modelamiento
Genesis	SI	>10	UNIX	NO	Gratis	Modelamiento
Dartnet	SI	<10	MAC	NO	Gratis	Modelamiento
SNNS	SI	>10	UNIX – WIN	SI	Gratis	Modelamiento
Aspirin/Mig	SI	>10	UNIX	SI	Gratis	Modelamiento
ALN Workbench	SI	<10	WIN	NO	Gratis	Modelamiento
PDP ++	SI	3	UNIX	NO	Gratis	Modelamiento
NeNet 1.0	SI	SOM	WIN	NO	Gratis	Modelamiento
Neural Networks at your Fingertips	NO	<10	WIN	NO	Gratis	Modelamiento
Neural Works	SI	>10	UNIX	NO	$14500	Varias
Neuro Forecaster	SI	NeuroFuzzy	WIN	NO	$1199	Pronósticos
Neuro Shell	SI	Predicción	WIN	NO	$395	Pronósticos
Neuro Solutions	SI	>10	WIN	SI	$1950	Varias
Hav Software	NO	FeedFoward	UNIX	NO	$50	Varias
Neuro Genetic Optimizar	SI	Predicción	WIN	NO	$195	Predicción
Neural Conection	SI	<10	WIN	NO	$995	Predicción
Pattern recognition PRO	SI	<10	WIN	SI	-	Patrones
Trajan Neural Simulator	SI	<10	WIN	NO	$995	Varias
Viscovery SOMine	SI	SOM	WIN	NO	$1495	Varias
Ainet	SI	<10	WIN	NO	$199	Modelamiento

Tabla 4.3.1

Donde:

Interfase Gráfica: Capacidad de ventanas en la interfase

Métodos: Tipos de redes que utiliza

Plataforma: Sistema Operativo sobre el cual opera la herramienta

Código: Si posee autogenerador de código

CAPÍTULO 5

5. DISEÑO E IMPLEMENTACIÓN DE FPSYS 1.0

5.1. ESPECIFICACIÓN FUNCIONAL

Este proyecto implementa una red neuronal que trata de simular un sistema que calcule el valor del factor P para los estudiantes de la Escuela Superior Politécnica del Litoral.

El sistema debe recibir un conjunto de datos que permita a la red entrenarse, de manera que pueda aprender a reconocer patrones y clasificar datos que sean ingresados posteriores al aprendizaje. Por lo tanto el sistema recibirá los datos a través de un archivo de Excel, estos serán procesados y la red podrá clasificar nuevos datos, los cuales también son esperados por el sistema por medio de un archivo.

El sistema generará los resultados y el usuario tendrá el control de validar estas respuestas a la red a través de la pantalla o por archivos.

El siguiente esquema representa las entradas y salidas del sistema implantado.

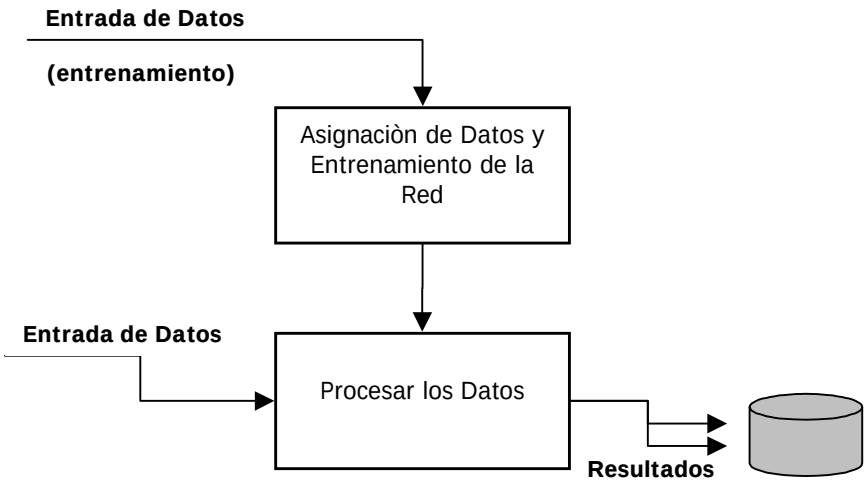


Figura 5.1.1

5.2. ANÁLISIS Y DISEÑO

Para ingresar los datos que servirán para el entrenamiento de la red, el sistema tomará esta información desde un archivo en Excel, el cual podrá ser especificado por el usuario del sistema. Entonces se producirá el entrenamiento de la red. Para este efecto se hace necesario el uso de estructuras de datos que almacenen la información ingresada. Se ha declarado un tipo de dato llamado FACTORP y PESO, estas estructuras son generadas con los datos almacenados en el archivo de Excel que se haya escogido para la fase de aprendizaje de la red.

Los datos ingresados en las estructuras pasan por los procesos de entrenamiento de la red, y los resultados son almacenados en una nueva estructura llamada CLASE. Este tipo de dato define las clasificaciones obtenidas luego del entrenamiento.

Para ingresar nuevos datos en la red con el propósito de obtener el factor P de dichos datos, el sistema proveerá la facilidad de escoger un nuevo archivo en Excel que contendrá los datos que se desean probar en la red. El sistema accesará al archivo no solo para tomar los datos estudiantiles sino también para almacenar los resultados generados por la red. Esta acción permitirá tomar los resultados y generar gráficos estadísticos.

Adicional a este archivo que contendrá los resultados de la red, también el sistema generará dos archivos de eventos que permitirán monitorear el desarrollo de la red en su etapa más critica, la fase de aprendizaje.

El sistema consta de tres componentes principalmente, los cuales describen procedimientos, procesos y presentación de información. En

cuanto a la presentación, existen Formas las cuales cumplen la función de crear la interface e interacción entre los usuarios y el sistema.

A través de esta interface se puede hacer uso de los principales procesos que se requieren para un ciclo de creación de una red neuronal, esto es, inicio, asignación de pesos, entrenamiento, actualización de valores y prueba de la red.

Estos procesos que posteriormente serán detallados utilizan ciertos procedimientos que dan la funcionalidad del caso. Estos procedimientos operan sobre los procesos que se requieren en la red neuronal y permiten manipular las estructuras de datos definidas. Adicionalmente, estos procedimientos leen y crean ciertos archivos para entrada y salida de información respectivamente.

Formas

El sistema pretende presentar una interface amigable que permita una eficaz interacción entre el usuario y los procesos. Se han implementado las siguientes formas:

Login

Esta forma cumple la función de Presentación del sistema y la validación de usuario. Si el usuario no se encuentra registrado para utilizar el sistema, esta forma no le permitirá acceder a la forma siguiente. Esta forma da acceso a la forma Main.

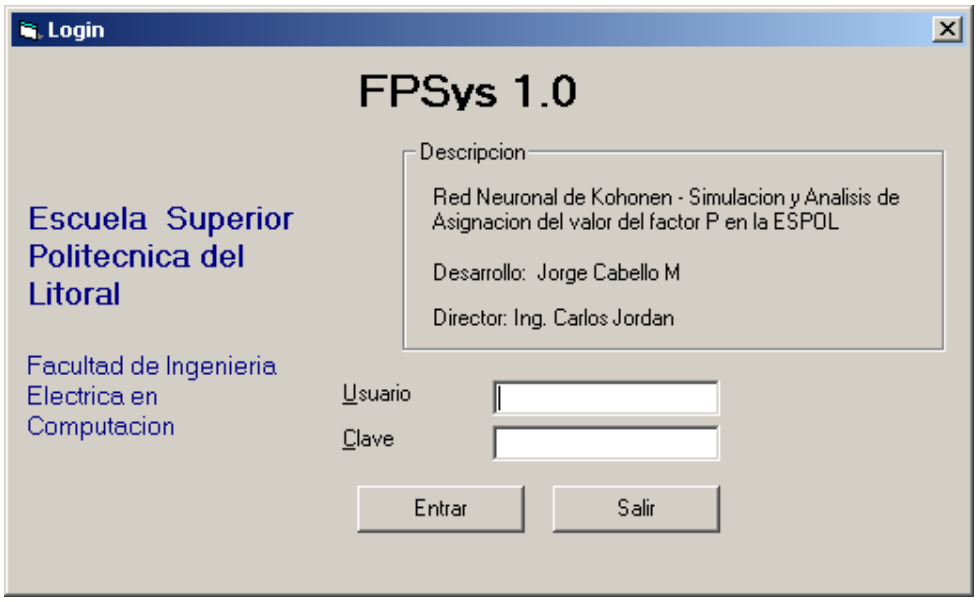


Figura 5.2.1

Main

Esta forma presenta la pantalla principal del sistema. De esta forma se toma control total sobre los procedimientos y procesos que involucran el ciclo de vida de una red neuronal, es decir, desde esta pantalla se puede inicializar y finalizar la red, entrenarla, configurar parámetros de entrenamiento, etc. Desde esta forma puedo ir a las formas: IniciarRed,

ProbarRed, ProbarArchivo, Estadisticas e Info. Adicionalmente, desde Main puedo acceder a visualizar los archivos de eventos almacenados durante el entrenamiento y prueba de datos.

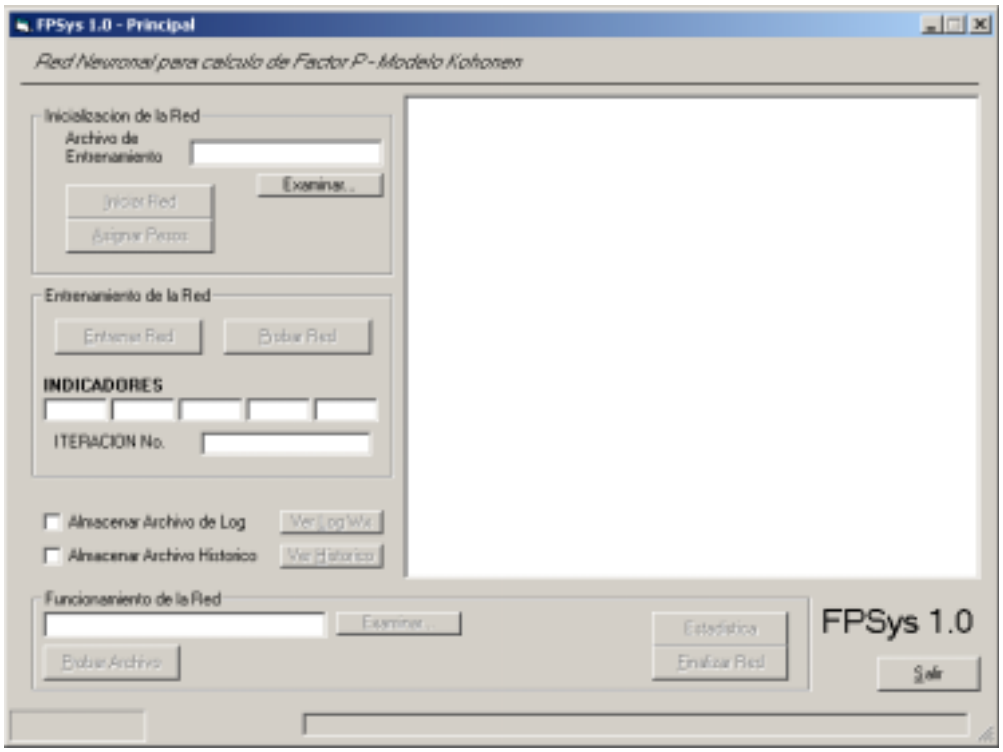


Figura 5.2.2

IniciarRed

Esta forma me permite configurar los parámetros de entrenamiento necesarios para ejecutar esta operación, esto es, las variables RACIMOS e ITERACIONES.

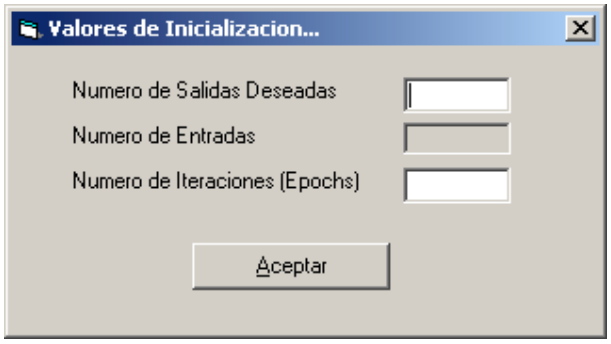


Figura 5.2.3

ProbarRed

Esta forma me permite ingresar datos referente a un elemento 0 persona e ingresarlo en la red neuronal, dando como resultado la clasificación obtenida de acuerdo al entrenamiento realizado.

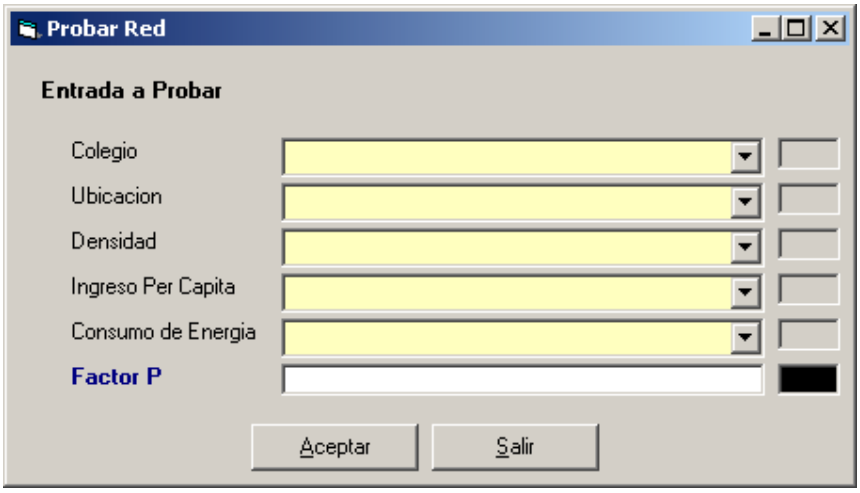


Figura 5.2.4

ProbarArchivo

En caso de necesitar clasificar más de un elemento en la red, existe la posibilidad de procesar todo un archivo en Excel el cual contenga los elementos a probar y sus datos.

Figura 5.2.5

Estadística

Esta forma presenta 2 gráficos estadísticos. Un gráfico estadístico que representa la variación de uno de los pesos en función del número de iteraciones. Este gráfico me permite mostrar en cual iteración los valores de los pesos se estabilizan.

El segundo gráfico representa la clasificación obtenida para cada uno de los datos procesados en la opción Probar Archivo.

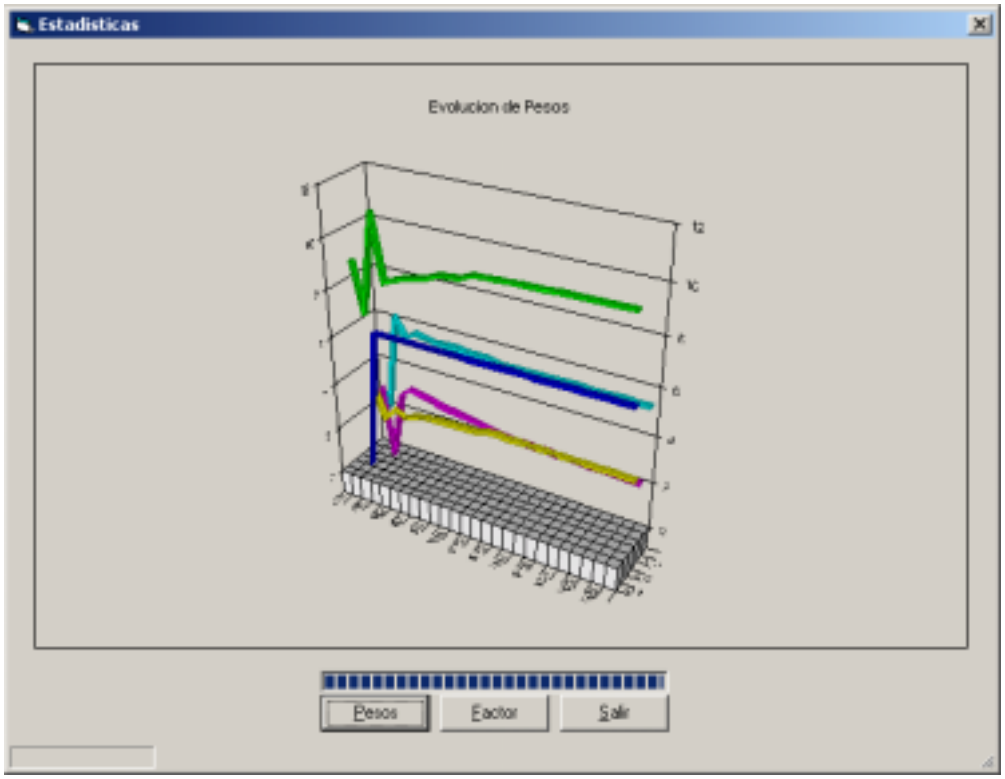


Figura 5.2.6

Info

Esta forma muestra información referente a la aplicación FPSYS 1.0

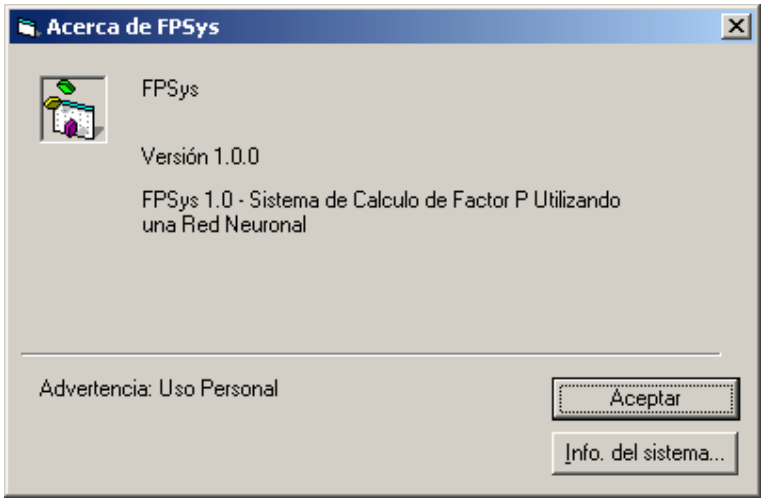


Figura 5.2.7

En el siguiente flujograma se muestra un diagrama de dependencias para las formas de tal manera que se visualice la navegación de la aplicación. De este gráfico podemos resumir los siguientes puntos:

1. Ingresar nombre del usuario y contraseña contra la aplicación.
2. Configurar la ubicación de los archivos temporales
3. Inicializar la red.
4. Ingresar valores de configuración para el entrenamiento a realizar.
5. Entrenar la red y existe la posibilidad de probar un elemento en la red ya entrenada.
6. Clasificar una gran cantidad de datos a través de un archivo.
7. Observar cuando se estabilizó la red y la clasificación de los datos
8. Revisar el log de eventos
9. Obtener información sobre la aplicación.

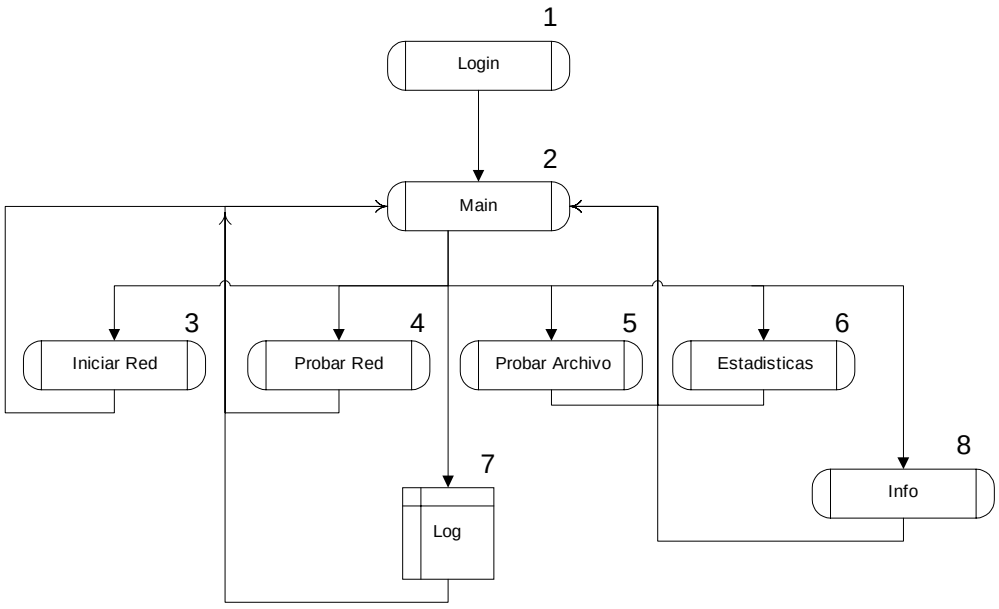


Figura 5.2.8

En cuanto a los procesos:

Como se mencionó en el temario anterior, la forma Main representa la interface principal donde interactúo con los procesos que intervienen en la creación de la red neuronal de Kohonen. Los procesos que son involucrados en la creación de la red son los siguientes:

Proceso 1: Iniciar la red.

Proceso 2: Asignar valores de configuración a la red.

Proceso 3: Entrenar la red.

Proceso 4: Probar los datos en la red entrenada.

Proceso 5: Finalizar la red.

Estos cinco procesos son secuenciales y suficientes para crear la red. Adicionalmente, se ha implementado varios procesos que me permite ver valores estadísticos y archivos de eventos, tales como:

Proceso 6: Ver log de eventos.

Proceso 7: Ver Archivo de históricos.

Proceso 8: Ver Gráficos Estadísticos.

Descripción de los procesos

Proceso Iniciar Red

Este proceso permite inicializar la red neuronal permitiendo ingresar los parámetros de configuración para el entrenamiento de la red. Este proceso crea también las estructuras de datos requeridas para las operaciones necesarias en los siguientes procesos, por lo tanto, mientras esta inicialización se haga satisfactoriamente, se estará en capacidad de entrenar la red. Es el primer proceso en el ciclo de la creación de la red y habilita usar el proceso AsignarPesos.

Proceso	Iniciar Red
Forma	FrmMain, frmlniciarRed
Dependencia	Ninguna
Parámetros Requeridos	Número de Racimos Número de Iteraciones en el aprendizaje Número de Entradas (constante) Nombre de Archivo a Entrenar
Objetivos	Crear estructuras de datos FACTORP Cargar archivo de datos en estructuras Configurar valores para entrenamiento

Proceso Asignar Pesos

Este proceso permite crear las estructuras de datos requeridas para almacenar los valores referentes a los pesos de las neuronas y generar aleatoriamente sus valores iniciales. Luego de haber generado estos

datos y haberlos cargado en la estructura PESOS, este proceso habilita la opción de entrenamiento de la red.

Proceso	Asignar Pesos
Forma	FrmMain
Dependencia	Iniciar_Pesos
Parámetros requeridos	Número de Racimos
Objetivos	Calcular estructuras de datos PESOS Generar valores iniciales/aleatorios Cargar archivo de datos en estructuras

Proceso Entrenar Red

Este proceso permite entrenar la red con los datos cargados en la estructura FACTORP y los pesos de la estructura PESOS, calculando la distancia euclidiana y utilizando los parámetros de configuración para generar los resultados requeridos. Al finalizar el proceso de entrenamiento, se crea una estructura llamada CLASE la cual almacena los valores finales de los pesos de las neuronas ganadoras. Esta estructura se utilizará para calcular la distancia euclidiana de nuevas neuronas que sean ingresadas a la red con el propósito de clasificar la neurona dentro de las clases obtenidas en el entrenamiento. Adicionalmente, en esta etapa se crean dos archivos de eventos que almacena el desarrollo de los pesos durante la fase de aprendizaje.

El archivo LOG.TXT almacena un muestro de los pesos durante el entrenamiento y el archivo LOGFULL.TXT almacena la historia completa de los pesos mientras se entrenaba la red. Estos archivos crecen de acuerdo al número de iteraciones requeridas en la forma frmlniciarRed, ya que este parámetro especifica la cantidad de veces que se repite el proceso de competencia en la red.

Proceso	Entrenar Red
Forma	FrmMain
Dependencia	Asignar Pesos
Parámetros requeridos	Número de Racimos Número de Iteraciones en el aprendizaje Archivos de Eventos
Objetivos	Calcular neuronas ganadoras(distancia) Crear estructuras de datos CLASE Carga neuronas ganadoras en CLASE

Proceso Probar Red

Este proceso permite ingresar los valores correspondientes al Factor P y clasificar esta entrada dentro de la clasificación obtenida en el proceso de entrenamiento utilizando el cálculo de la distancia entre la neurona ingresada y las neuronas en la estructura CLASE. Es decir, en este momento se puede ingresar los datos de un estudiante en la red y solicitar al sistema su factor P, de acuerdo a los valores obtenidos durante la fase de aprendizaje. Este proceso ha sido diseñado para ingresar una sola neurona a la vez en la red a través de la forma frmProbarRed.

Proceso	Probar Red
Forma	FrmMain, frmProbarRed
Dependencia	Entrenar Red
Parámetros Requeridos	Colegio Ubicación Domiciliaria Ingreso Per Cápita Consumo de Energía Eléctrica Densidad Poblacional Datos almacenados en CLASE
Objetivos	Ingresar datos de un estudiante Clasificar al estudiante Mostrar factor P del estudiante

Proceso Probar Archivo

Tiene la misma función de Probar Red, pero en este proceso se puede clasificar más de un estudiante a la vez. Luego de haber entrenado la red, el usuario puede clasificar una cantidad determinada de datos contenidos en un archivo en Excel. El proceso es muy similar al anterior: Calcula la distancia euclidiana de la neurona a probar con las neuronas almacenadas en CLASE. Este cálculo lo hace para cada una los estudiantes (datos) contenidos en el archivo especificado para la clasificación. La clasificación o Factor P obtenido por cada uno de los estudiantes es almacenado en el mismo archivo utilizado para ingresar los datos a la red, el cual puede ser accesado normalmente luego de la clasificación.

Proceso	Probar Archivo

Forma	FrmMain
Dependencia	Entrenar Red
Parámetros Requeridos	Archivo de Excel para clasificar Datos almacenados en CLASE Pesos almacenados en PESOS
Objetivos	Ingresar archivo a clasificar Clasificar los estudiantes Almacenar el Factor P de los estudiantes

Proceso Estadísticas

Este proceso permite tomar los datos obtenidos en una clasificación previa y formar dos gráficos estadísticos de la red. Estos datos han sido almacenados en un archivo de Excel durante el proceso Probar Archivo, los cuales son tomados para generar los gráficos.

Luego de que los datos han sido recopilados, se muestra un visor para los dos gráficos:

- Grafico de Pesos
- Gráfico de Factor P.

Gráfico de Pesos: Este permite ver el desarrollo de los pesos en función del número de iteraciones. Debido a que el número de pesos es ilimitado, se escoge una neurona al azar y es monitoreada y se gráfica su comportamiento durante la etapa de entrenamiento. De esta manera

podemos visualizar en que iteración los pesos se estabilizan y permanecen constantes.

Gráfico de Factor P: Este permite visualizar la clasificación obtenida durante la etapa de ingreso de nuevos datos a la red, de manera que, con este gráfico puedo saber cuantos elementos o estudiantes obtuvieron un determinado Factor P.

Proceso	Estadísticas
Forma	FrmMain, frmEstadìsticas
Dependencia	Probar Archivo, Entrenar Red
Parámetros Requeridos	Ninguno
Objetivos	Mostrar estabilización de la red Clasificar estudiantes Mostrar Grafico de Clasificación

Proceso Finalizar

Este proceso permite liberar las estructuras de datos que contienen los datos de los estudiantes y los pesos asignados a cada neurona en la red con el propósito de crear una nueva red e iniciar el entrenamiento con los nuevos valores.

Proceso	Finalizar Red
Forma	FrmMain
Dependencia	Entrenar Red
Parámetros Requeridos	Ninguno

Objetivos	Finalizar la red Liberar estructuras de datos Habilitar nuevo entrenamiento

Proceso Ver Log

Ver Log permite abrir el archivo de eventos almacenado en el proceso de entrenamiento, el cual posee el desarrollo de los pesos de una neurona tomada al azar, con el propósito de evaluar en que momento o iteración se estabiliza el proceso de aprendizaje. Este archivo se lo ha llamado LOG.TXT y debido a que es un archivo temporal, se guarda en un directorio temporal y es visualizado desde el Visor de Texto de la forma frmMain.

Proceso	Ver Log
Forma	FrmMain
Dependencia	Entrenar Red
Parámetros Requeridos	Pesos almacenados en PESOS
Objetivos	Ver desarrollo de un peso durante fase de aprendizaje

Proceso Ver Histórico

Ver Histórico permite ver el archivo de eventos completo almacenado durante el proceso de aprendizaje de la red. Este guarda el desarrollo de todos los pesos de todas las neuronas de la red en todas y cada una de las iteraciones durante el entrenamiento. Este archivo se lo ha llamado LOGFULL.TXT y también es almacenado en un directorio temporal y es visualizado desde el utilitario WRITE de Windows.

Proceso	Ver Històrico
Forma	FrmMain
Dependencia	Entrenar Red
Parámetros Requeridos	Pesos almacenados en PESOS
Objetivos	Ver historia de los pesos durante fase de aprendizaje

En el siguiente gráfico, se muestra el flujo de los procesos durante todo el ciclo de una red neuronal creada en el sistema.

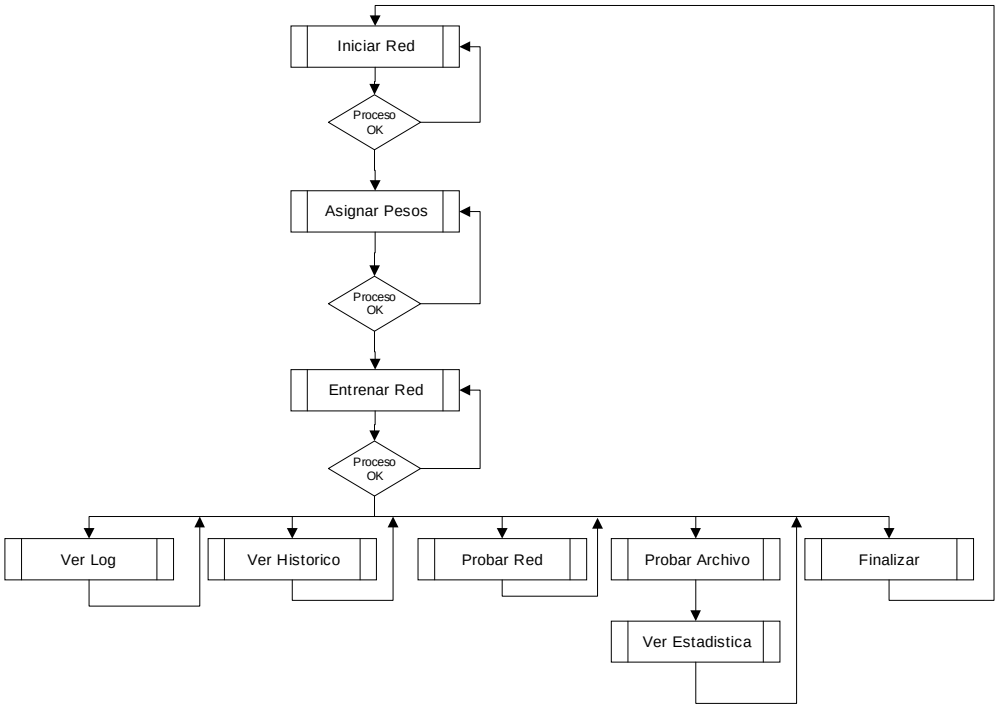


Figura 5.2.9

Entrando en la programación del sistema, se han implementado 8 módulos desarrollados desde Visual Basic 6.0 con la finalidad de ejecutar

tareas específicas dentro de los procesos mencionados en las páginas anteriores. Estos módulos son los siguientes:

Modulo General

Establece todas las variables globales del sistema y declara las estructuras de PESOS, FACTORP y CLASE utilizadas para el entrenamiento y prueba de la red.

Modulo Calcular_Distancia_Eucl()

Procedimiento llamado por el proceso Entrenar Red, Probar Red y Probar Archivo. Se encarga de calcular la distancia euclidiana entre el valor de la neurona que ingresa al entrenamiento y los pesos almacenados en la red. También determina cuales de los pesos son los más cercanos a la neurona, estimando así que neurona ha sido activada y cual es el ganador.

Modulo Actualizar_Pesos ()

Procedimiento llamado por el proceso Entrenar Red. Se encarga de tomar la neurona ganadora y actualizar sus pesos con la función de actualización mencionada en el capítulo 3 de esta tesis.

Modulo Actualizar_Ganador ()

Procedimiento llamado por el proceso Entrenar Red. Se encarga de guardar las neuronas ganadoras en la estructura CLASE con el propósito de utilizarlas al momento de probar una nueva neurona en la red.

Modulo Probar_Archivo ()

Procedimiento llamado por el proceso Probar Archivo. Se encarga de acceder al archivo en Excel para ingresar nuevos datos a la red y clasificarlos de acuerdo a las clases obtenidas en la fase de aprendizaje de la red.

Modulo Cerrar_logs ()

Procedimiento llamado por el proceso Asignar Pesos. Se encarga de verificar que los archivos de eventos se encuentren cerrados, lo cual asegura que los archivos temporales de eventos generados en el nuevo entrenamiento representen los datos de la nueva red.

Modulo Generar_Tabla_Resultados ()

Procedimiento llamado por el proceso Estadísticas. Se encarga de tomar los resultados generados al probar un grupo de estudiantes en la red ya entrenada y generar una tabla con la clasificación obtenida, utilizada posteriormente para crear un gráfico estadístico. Estos resultados representan el factor P de cada uno de los estudiantes.

Modulo Borrar_Indicadores ()

Procedimiento llamado por el proceso Iniciar Red. Se encarga de borrar los valores de los indicadores de datos. El propósito de estos indicadores es mostrar los datos del estudiante actual en la fase de entrenamiento.

En cuanto a los archivos que se necesitan utilizar, el sistema se basa en un sistema de información de lectura y escritura de archivos específicos, los cuales contienen datos, resultados y eventos resultantes de la operación de la red neuronal. Estos archivos tienen la siguiente estructura y descripción:

Archivo de Datos para entrenamiento de la red

Este archivo de Excel permite tomar datos correspondientes a un muestreo de la población referente a las variables o criterios bajo los cuales se toma el valor del factor P, tal como el Ingreso Per Cápita de la familia del estudiante y entrenar la red. Para el correcto funcionamiento de la red y por la consistencia de los datos resultantes, este archivo necesariamente debe tener la siguiente estructura:

COL	UBI	DEN	IPC	CEE
4	10	4	2	5

12	12	3	0	2
7	10	4	3	6
9	3	3	3	3
4	2	4	3	1
9	10	3	4	6
8	6	2	6	7
5	10	3	4	7

Donde,

- La primera columna (COL) representa el Colegio del estudiante
- La segunda columna (UBI) representa la Ubicación domiciliaria del estudiante
- La tercera columna (DEN) representa la Densidad Poblacional de la vivienda del estudiante
- La cuarta columna (IPC) representa el Ingreso per Cápita de la familia del estudiante
- La quinta columna (CEE) representa el Consumo de energía Eléctrica de la vivienda estudiantil

Este archivo puede estar localizado en cualquier partición de un disco local.

Archivo de Datos para ingresar nuevos estudiantes

Este archivo de Excel permite tomar valores de nuevos estudiantes e ingresarlos a la red ya entrenada. Mantiene la misma estructura que el archivo de datos de entrenamiento. En este archivo también se guardan los resultados o clasificaciones obtenidas en la red luego de ingresar los datos y obtener el factor P para cada estudiante. Este archivo puede estar localizado en cualquier partición de un disco local.

Archivo de Eventos

Este archivo almacena un muestro de los valores de una neurona cada 10 iteraciones, teniendo así una manera de determinar cuando el proceso de entrenamiento se estabiliza. Este archivo es almacenado en formato texto con el nombre LOG.TXT en el directorio temporal, el cual está localizado en la carpeta c:\fpsys\logs debe ser configurado en la pantalla principal del sistema. Este archivo mantiene el siguiente formato:

Iterac.1 W1(2,9,3,3,0)
Iterac. 11 W1(1.45, 8.45, 2.45, 2.55, 0)
Iterac. 21 W1(1.475, 8.475, 2.475, 2.525, 0)
Iterac. 31 W1(1.483333, 8.483334, 2.483333, 2.516667, 0)
Iterac. 41 W1(1.4875, 8.4875, 2.4875, 2.5125, 0)
Iterac. 51 W1(1.49, 8.49, 2.49, 2.51 , 0)
Iterac. 61 W1(1.491667, 8.491667, 2.491667, 2.508333, 0)
Iterac. 71 W1(1.492857, 8.492857, 2.492857, 2.507143, 0)

Iterac. 81 W1(1.49375, 8.49375, 2.49375, 2.50625, 0)

Iterac. 91 W1(1.494444, 8.494445, 2.494444, 2.505556, 0)

Archivo Histórico

Este archivo guarda el desarrollo de todos los pesos de todas las neuronas de la red en todas y cada una de las iteraciones durante el entrenamiento. Este archivo se lo ha llamado LOGFULL.TXT y también es almacenado en el directorio de archivos temporales y es visualizado desde el utilitario WRITE de Windows.

Por ejemplo para los pesos del primer dato se desarrollará de la siguiente manera:

Iteración No. 1 ----------

W 1 (2, 9, 3, 3, 0)

Iteración No. 2 ----------

W 1(1, 8, 2, 3, 0)

Iteración No. 3 ----------

W 1 (1.25, 8.25, 2.25, 2.75, 0)

Iteración No. 4 ----------

W 1 (1.333333, 8.333333, 2.333333, 2.666667, 0)

Iteración No. 5 ----------

W 1 (1.375, 8.375, 2.375, 2.625, 0)

Iteración No. 6 ----------

W 1(1.4, 8.4, 2.4, 2.6, 0)

Iteración No. 7 ----------

W 1 (1.416667, 8.416667, 2.416667, 2.583333, 0)

Archivo de Estabilización de Pesos

Este archivo almacena un registro de un peso aleatorio durante la fase de entrenamiento con el propósito de tomar esta tabla y generar un gráfico estadístico de la red. Este archivo se almacena en el directorio de archivos temporales con el nombre PESOS.XLS. Su estructura es la siguiente:

PesoCol	PesoUbi	PesoDen	PesoIpc	PesoCee
9	0	3	3	2
8	0	2	3	1
8.25	0	2.25	2.75	1.25
8.33333	0	2.33333	2.666667	1.333333
8.375	0	2.375	2.625	1.375
8.4	0	2.4	2.6	1.4
8.416667	0	2.416667	2.583333	1.416667
8.428572	0	2.428571	2.571429	1.428571

5.3. INSTRUMENTACIÓN

5.3.1. ESTRUCTURAS DE DATOS

El sistema requiere 3 estructuras de datos, las cuales almacenan estrictamente datos de entrenamiento y resultados de ¡a red neuronal. Estas estructuras son las siguientes:

Estructura: PESOS

Un Arreglo de m-elementos donde m es definido por ¡a cantidad de racimos o clases que se deseen entrenar. Cada elemento del arreglo, posee a su vez, 6 campos que guardan las variables de peso:

PesoCol: Ponderación por Colegio

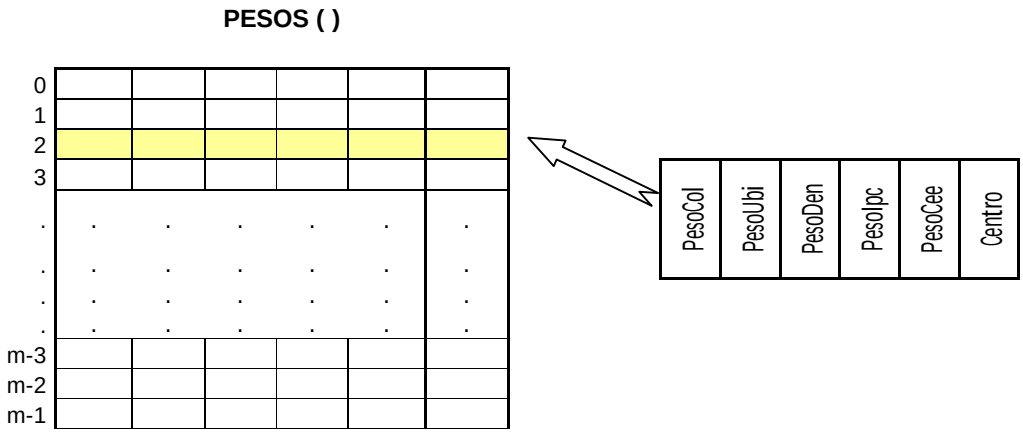
PesoUbi: Ponderación por Ubicación domiciliaria

PesoDen: Ponderación por Densidad Poblacional

PesolPc: Ponderación por Ingreso Per Cápita

PesoCee: Ponderación por Consumo de energía Eléctrica

Centro: Ponderación alcanzado por racimo



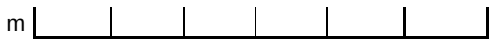


Figura 5.3.1

Estructura: FACTOR

Un arreglo de n elementos, donde n es definido por la cantidad de valores o datos que se desean entrenar o probar en el sistema. Cada elemento contiene 6 campos que guardan los valores correspondientes a los datos de colegio, Ubicación domiciliaria, Densidad, Ingreso per Cápita y Consumo de Energía Eléctrica.

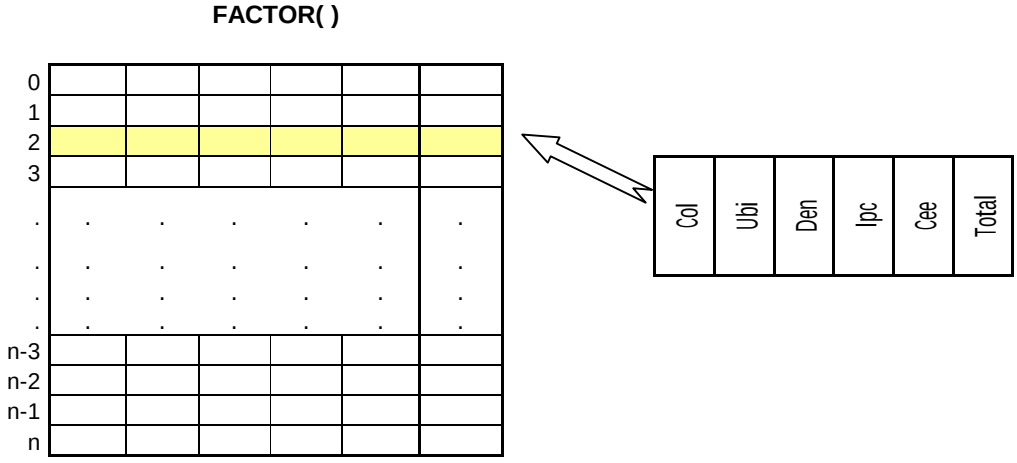


Figura 5.3.2

Estructura: CLASE

Un arreglo de igual dimensionamiento que el arreglo tipo PESO. En este arreglo se almacenan los datos correspondientes a las clases o

clasificación generadas al finalizar la etapa de entrenamiento de la red. Estos valores son utilizados para compararlos con los datos a probar, utilizando la distancia euclidiana para clasificar los datos resultantes.

5.3.2. VARIABLES

En el sistema se han definido las siguientes variables globales:

PESOS

Tipo: PESO

Descripción: Arreglo de almacenamiento de PESOS

CLASE

Tipo: CLASE

Descripción: Arreglo de almacenamiento de CLASE o Centro de Racimos

FACTORP

Tipo: FACTOR

Descripción: Arreglo de almacenamiento de DATOS

RACIMOS

Tipo: Entero

Descripción: Cantidad de racimos a clasificar. Si deseo entrenar la red con un rango de Factor P de 3 a 40, entonces, RACIMOS es la cantidad de clasificaciones que puedo obtener. En este ejemplo RACIMOS es igual a 40-3=38

Número_Datos

Tipo: Entero

Descripción: Cantidad de datos a entrenar o probar en la red

DISTANCIA

Tipo: Precisión Simple

Descripción: Distancia entre i-neurona y k-peso.

Donde: $número\_de\_datos \geq i \geq 1$

$Cantidad\_de\_pesos \geq k \geq 1$

Minima_distancia

Tipo: Precisión Simple

Descripción: Distancia mínima de las DISTANCIAS tomadas

Ganador

Tipo: Entero

Descripción: i-dato ganador en la competencia por activación

Iteraciones

Tipo: Entero

Descripción: Cantidad de veces que se desea repetir el proceso de entrenamiento por dato

FILELOG

Tipo: Cadena

Descripción: Almacena cadena de caracteres para enviar a log de eventos

5.3.3. ALGORITMOS

Declaración de Variables Globales y **Constantes**

Se definen las siguientes variables y valores:

- fMainForm As frmMain
- Archivo como Cadena
- ENTRADAS como entero de valor 5
- Path_Archivo_Entrenamiento como cadena
- Path_Archivo_Resultados como cadena

- BAND como entero
- RACIMOS como entero
- FILELOG como cadena
- DISTANCIA como número real de simple precisión
- Minima_distancia como número real de simple precisión
- Ganador como número real de simple precisión
- Número_Datos como entero
- Iteraciones como entero

Declaraciones de Estructuras de Datos

Tipo CLASE

PesoCol como número real de simple precisión
PesoUbi como número real de simple precisión
PesoDen como número real de simple precisión
Pesolpc como número real de simple precisión
PesoCee como número real de simple precisión
Centroide como número real de simple precisión

Variable CLASE como tipo CLASE

Tipo PESOS

PesoCol como número real de simple precisión
PesoUbi como número real de simple precisión
PesoDen como número real de simple precisión
Pesolpc como número real de simple precisión
PesoCee como número real de simple precisión
Centro como número real de simple precisión

Variable PESO() como tipo PESOS

```
Tipo FACTOR
        COL como número real de simple precisión
        UBI como número real de simple precisión
        DEN como número real de simple precisión
        IPC como número real de simple precisión
        CEE como número real de simple precisión
        Total como número real de simple precisión
Variable FACTORP como tipo FACTOR
```

Proceso Inicializar Red

Inicializa la red, creando el arreglo que contiene los datos a entrenar

Crea Objeto tipo Excel

Abre archivo a entrenar

Calcula la cantidad de datos a entrenar

Carga valores del archivo Excel al arreglo FACTORP

Cierra el archivo de Excel

Borrar_Indicadores

Llama a forma para ingresar valores de entrenamiento

frmIniciarRed.Show

Fin de procedimiento

Proceso Asignar Pesos

Asigna pesos aleatorios y los carga en un arreglo PESO

Cierra archivo de logs si están abiertos

Redimensiona arreglos de acuerdo a los racimos o cluster

Ingresados en frmIniciarRed

Genera valores aleatorios para los pesos y los carga en el arreglo PESO

Fin de procedimiento

Proceso Entrenar Red

Entrena la red con el archivo seleccionado y los parámetros cargados en la frmIniciarRed

Crea Objeto tipo Excel

Abre archivos de log para almacenar históricos y eventos

Procesa los datos de acuerda al número de iteraciones solicitadas

Guarda valores de actualizaciones de pesos en archivo histórico Guarda valor de actualizaciones de un peso ejemplo en archivo de log

Por cada dato

Borrar_Indicadores(FACTORP(k) .COL, FACTORP(k).UBI, — FACTORP(k).DEN, FACTORP(k).IPC, FACTORP(k).CEE)

Por cada racimo

Calcula distancia euclidiana

Calcula ganador por competencia

Actualiza los pesos del ganador

Clasifica los ganadores en un arreglo final CLASE

Llama a módulo Actualizar_Ganador(i)

Cierra archivos de log y Excel

Fin de procedimiento

Proceso Probar Archivo

Prueba archivo de datos en red entrenada

Llama a procedimiento para probar datos en la red

Llama a forma frmProbarArchíVO

Fin de procedimiento

Proceso Estadísticas Pesos

Muestra Gráfico correspondiente al desarrollo y actualización de pesos

Acceso a archivo Pesos.xls

Defino tipo de gráfico

Genera Arreglo con datos de los pesos

Genera gráfico a través del arreglo anterior

Cierro archivo Pesos.xls

Fin de Procedimiento

Proceso Estadísticas Pesos

Muestra Gráfico correspondiente al resultado de los racimos generados en una prueba de datos

Llama a módulo que ordena los datos y los clasifica

Genera_Tabla_Resultados

Abre objeto de Excel

Cargo datos ordenados y clasificados en arreglo

Genero gráfico con los datos del arreglo generado

Cierro objeto Excel

Fin de Procedimiento

Proceso Probar Archivo

Prueba datos desde un archivo en Excel en la red entrenada

Crea Objeto tipo Excel

Abre archivo a probar

Abre archivo de Excel que contiene los datos a probar

Calcula la cantidad de datos a probar

Redimensiona el arreglo FACTORP a la cantidad de datos requerida

Carga valores de datos del archivo de Excel al Arreglo FACTORP

Prueba los datos en la red

Calcula distancia

Calcula ganador por competencia

Llama a módulo Borrar_Indicadores()

Almacena resultado en una columna del archivo de Excel

Cierra archivo de Excel

Fin de Procedimiento

Proceso Actualizar Pesos

Actualiza los pesos en la neurona ganadora

Actualizo pesos con fórmula

PESO(ganador) = PESO(ganador) + ((1/*t)* * (FACTORP(k) - PESO(Ganador)))

Fin de Proceso

5.4. CODIFICACIÓN DE ALGORITMOS

```
Option Explicit
'Declaración de constantes y Variable Globales del Sistema*
'*****************************************************************
   Public fMainForm As frmMain
   Public Archivo As String
   Const ENTRADAS As Integer = 5
   Public Path_Archivo_Entrenamiento As String
   Public Path_Archivo_Resultados As String
   Public BAND As Integer
   Public RACIMOS As Integer
   Public FILELOG As String
   Public DISTANCIA As Single
   Public Minima_distancia As Single
   Public Ganador As Single
   Public Numero_Datos As Integer
   Public Iteraciones As Integer


'Declaraciones de Estructuras de Datos
'*****************************************************************
   Public Type CLASE
      PesoCol As Single
      PesoUbi As Single
      PesoDen As Single
      PesoIpc As Single
      PesoCee As Single
      Centroide As Integer
   End Type
   Public CLASE() As CLASE

   Public Type PESOS
      PesoCol As Single
      PesoUbi As Single
      PesoDen As Single
      Pesolpc As Single
      PesoCee As Single
      Centro As Single
   End Type
   Public PESO() As PESOS
```

```
    Public Type FACTOR
        COL As Single
        UBI As Single
        DEN As Single
        IPC As Single
        CEE As Single
        Total As Single
    End Type
    Public FACTORP() As FACTOR


Sub main()
'*************************************************************************
    Dim flogin As New frmLogin
    flogin.Show vbModal
    Unload flogin
    frmMain.Show
End Sub



Sub Probar_Archivo(Path As String)
'Prueba datos desde un archivo en Excel en la red entrenada
'*****************************************************************
    frmMain.StatusBar1.SimpleText = "Probando datos en la red..."
    Dim contador As Integer
    Dim j, k, UltimoDato As Integer
    Dim dato_procesado As String
    Dim ApExcel As Excel.Application
    Set ApExcel = CreateObject("Excel.Application")
    UltimoDato = 0
    frmMain.ProgressBar1.Min = 0
    frmMain.ProgressBar1.Value = 0
    CrLf$ = Chr(13) & Chr(10)
    FILELOG = "Procesando archivo de datos en la red... "
    frmMain.txtVisor.Text = frmMain.txtVisor.Text & FILELOG & CrLf$
    'Abre archivo de Excel que contiene los datos a probar
    ApExcel.Workbooks.Open (Path)
    ApExcel.Visible = False
    contador = 2
    'Calcula la cantidad de datos a probar
    Do While ApExcel.Cells(contador, 1) <> ""
        UltimoDato = contador
        contador = contador + 1
    Loop
    Numero_Datos = UltimoDato - 1
    'Redimensiona el arreglo FACTORP a la cantidad de datos requerida
    frmMain.ProgressBar1.Max = Numero_Datos
```

```
ReDim FACTORP(UltimoDato - 1)
i = 1
j = 2
'Carga valores de datos del archivo de Excel al Arreglo FACTORP
Do
   FACTORP(i).COL = ApExcel.Cells(j, 1)
   FACTORP(i).UBI = ApExcel.Cells(j, 2)
   FACTORP(i).DEN = ApExcel.Cells(j, 3)
   FACTORP(i).IPC = ApExcel.Cells(j, 4)
   FACTORP(i).CEE = ApExcel.Cells(j, 5)
   FACTORP(i).Total = 0
   i = i + 1
   j = j + 1
Loop While i < UltimoDato
i = 1
k = 1
j = 2
'Prueba los datos en la red
Do
   For i = 1 To RACIMOS
      'Calcula distancia
      DISTANCIA = 0
      DISTANCIA = DISTANCIA + (Abs(CLASE(i).PesoCee - FACTORP(k).CEE)) ^ 2
      DISTANCIA = DISTANCIA + (Abs(CLASE(i).PesoCol - FACTORP(k).COL)) ^ 2
      DISTANCIA = DISTANCIA + (Abs(CLASE(i).PesoDen - FACTORP(k).DEN)) ^ 2
      DISTANCIA = DISTANCIA + (Abs(CLASE(i).PesoIpc - FACTORP(k).IPC)) ^ 2
      DISTANCIA = DISTANCIA + (Abs(CLASE(i).PesoUbi - FACTORP(k).UBI)) ^ 2
      'Calcula ganador por competencia
      Call Borrar_Indicadores.Borrar_Indicadores(FACTORP(k).COL,
      FACTORP(k).UBI, _
      FACTORP(k).DEN, FACTORP(k).IPC, FACTORP(k).CEE, FACTORP(k).Total)
      If i = 1 Then
         Ganador = i
         Minima_distancia = DISTANCIA
      Else
         If DISTANCIA < Minima_distancia Then
            Ganador = i
            Minima_distancia = DISTANCIA
         End If
      End If
   Next i
   dato_procesado = CLASE(Ganador).Centroide
   'Almacena resultado en una columna del archivo de Excel
   ApExcel.Cells(j, 6) = dato_procesado
   j = j + 1
   k = k + 1
   frmMain.ProgressBar1.Value = k - 1
Loop While k < UltimoDato
```

```
    'Cierra archivo de Excel
    ApExcel.ActiveWorkbook.Save
    ApExcel.Workbooks.Close
    ApExcel.Quit
    Set ApExcel = Nothing
    FILELOG = "Datos Procesados"
    frmMain.txtVisor.Text = frmMain.txtVisor.Text & UltimoDato & " " & FILELOG &
CrLf$
    frmMain.txtVisor.Text = frmMain.txtVisor.Text & _
    " Las respuestas se han grabado en el Archivo " & frmMain.txtProbar.Text & CrLf$
End Sub



Sub Generar_Tabla_Resultados()
'Ordena y Clasifica resultados de archivo
'*********************************************************************
    Dim Dato_Repetido As Boolean, i As Integer, j As Integer, Suma As Integer
    Dim Dato As Integer, Dato_Variable As Integer
    Dim ApExcel As Excel.Application
    Dim ApBook As Excel.Workbook
    Dim ApSheet As Excel.Worksheet
    'Accesa a archivo de Excel que contiene datos de resultados
    Set ApExcel = CreateObject("Excel.Application")
    Set ApBook = ApExcel.Workbooks.Open(frmMain.txtProbar)
    Set ApSheet = ApBook.Worksheets(1)
    ApExcel.Visible = True
    ApSheet.Range("F" & 2, "F" & (Numero_Datos + 1)).Copy
    Set ApSheet = ApBook.Worksheets(2)
    ApSheet.Range("C" & 1, "C" & (Numero_Datos)).PasteSpecial
    'Ordena datos en forma ascendente
    ApSheet.Range("C" & 1, "C" & (Numero_Datos)).SortSpecial (xlAscending)
    i = 2
    'Clasifica los datos ordenados de acuerdo a los racimos requeridos inicialmente
    Do
        DatoRepetido = False
        Suma = 1
        Dato_Variable = i + 1
        Dato = ApSheet.Cells(i, 3)
        Do While Dato = ApSheet.Cells(Dato_Variable, 3)
            Suma = Suma + 1
            Dato_Variable = Dato_Variable + 1
            Dato_Repetido = True
        Loop
        For j = 1 To RACIMOS
            If Dato = j Then ApSheet.Cells(j, 2) = Suma
        Next j
        If Dato_Repetido = False Then
```

```
            i = i + 1
        Else
            i = Dato_Variable
        End If
    Loop While i <= Numero_Datos
    'Almacena clasificadón en archivo de Excel
    'Esta ínformadón es requerida para generar gráfico de racimos
    ApExcel.ActiveWorkbook.Save
    ApExcel.Workbooks.Close
    ApExcel.Quit
    Set ApExcel = Nothing
End Sub



Sub Cerrar_logs()
'Cierra archivos de log abiertos
'*************************************************************
    If BAND = 1 Then
        AppActivate "logfull - WordPad"
        SendKeys "%{F4}", True
    End If
End Sub



Sub Calcular_Distancia_Eucl(ByVal i As Integer, ByVal k As Integer)
'Calcula Distancia Euclideana
'*********************************************************************
    DISTANCIA = 0
    DISTANCIA = DISTANCIA + (Abs(PESO(i).PesoCee - FACTORP(k).CEE)) ^ 2
    DISTANCIA = DISTANCIA + (Abs(PESO(i).PesoCol - FACTORP(k).COL)) ^ 2
    DISTANCIA = DISTANCIA + (Abs(PESO(i).PesoDen - FACTORP(k).DEN)) ^ 2
    DISTANCIA = DISTANCIA + (Abs(PESO(i).Pesolpc - FACTORP(k).IPC)) ^ 2
    DISTANCIA = DISTANCIA + (Abs(PESO(i).PesoUbi - FACTORP(k).UBI)) ^ 2
End Sub



Public Sub Borrar_Indicadores(ByVal i, j, k, l, m, n)
'Borrar indicadores
'*********************************************************************
    frmMain.COL.Caption = i
    frmMain.UBI.Caption = j
    frmMain.DEN.Caption = k
    frmMain.IPC.Caption = l
    frmMain.CEE.Caption = m
    frmMain.lbliteracion.Caption = n
```

```
    frmMain.COL.Refresh
    frmMain.UBI.Refresh
    frmMain.DEN.Refresh
    frmMain.IPC.Refresh
    frmMain.CEE.Refresh
    frmMain.lbliteracion.Refresh
End Sub


Sub Actualizar_Pesos(ByVal t As Integer, ByVal k As Integer)
'Actualíza los pesos en la neurona ganadora
'*************************************************************************
    PESO(Ganador).PesoCee = PESO(Ganador).PesoCee + ((1 / t) * _
    (FACTORP(k).CEE - PESO(Ganador).PesoCee))
    PESO(Ganador).PesoCol = PESO(Ganador).PesoCol + ((1 / t) * _
    (FACTORP(k).COL - PESO(Ganador).PesoCol))
    PESO(Ganador).PesoDen = PESO(Ganador).PesoDen + ((1 / t) * _
    (FACTORP(k).DEN - PESO(Ganador).PesoDen))
    PESO(Ganador).Pesolpc = PESO(Ganador).Pesolpc + ((1 / t) * _
    (FACTORP(k).IPC - PESO(Ganador).Pesolpc))
    PESO(Ganador).PesoUbi = PESO(Ganador).PesoUbi + ((1 / t) * _
    (FACTORP(k).UBI - PESO(Ganador).PesoUbi))
    PESO(Ganador).Centro = FACTORP(k).Total
End Sub


Sub Actualizar_Ganador(ByVal i As Integer)
'Actualiza la neurona ganadora en arreglo de resultados
'*************************************************************************
    CLASE(i).PesoCee = PESO(i).PesoCee
    CLASE(i).PesoCol = PESO(i).PesoCol
    CLASE(i).PesoUbi = PESO(i).PesoUbi
    CLASE(i).PesoDen = PESO(i).PesoDen
    CLASE(i).PesoIpc = PESO(i).Pesolpc
    CLASE(i).Centroide = PESO(i).Centro
End Sub


Private Sub cmdexaminar_click()
'*************************************************************************
    cdiag1.ShowOpen
    txtPath.Text = cdiag1.FileName
    cmdInicializarRed.Enabled = True
End Sub
```

```
Private Sub cmdAsignarPesos_Click()
'Asigna pesos aleatorios y los carga en un arreglo PESO
'*************************************************************************
    Dim i As Integer
    StatusBar1.SimpleText = "Asignando Pesos Iniciales..."
    i = 1
    'Cierra archivo de logs si están abiertos
    Call Cerrar_logs.Cerrar_logs
    ProgressBar1.Max = RACIMOS
    ProgressBar1.Value = 0
    CrLf$ = Chr(13) & Chr(10)
    FILELOG = "Asignando Pesos Aleatorios..."
    txtVisor.Text = txtVisor.Text & FILELOG & CrLf$
    txtVisor.Refresh
    'Redimensiona arreglos de acuerdo a los racimos o cluster
    'ingresados en frmIniciar Red
    ReDim PESO(RACIMOS)
    'Genera valores aleatorios para los pesos y los carga en el arreglo PESO
    Do
       PESO(i).PesoCol = Int(12 * Rnd + 1)
       PESO(i).PesoDen = Int(4 * Rnd + 1)
       Do
          PESO(i).PesoCee = Int(6 * Rnd + 1)
       Loop While PESO(i).PesoCee = 4
       Do
          PESO(i).Pesolpc = Int(5 * Rnd)
       Loop While PESO(i).Pesolpc = 1
       Do
          PESO(i).PesoUbi = Int(15 * Rnd)
       Loop Until PESO(i).PesoUbi = 0 Or PESO(i).PesoUbi = 2 _
       Or PESO(i).PesoUbi = 6 Or PESO(i).PesoUbi = 10 Or _
       PESO(i).PesoUbi = 12 Or PESO(i).PesoUbi = 14
       PESO(i).Centro = 0
       i = i + 1
       ProgressBar1.Value = i - 1
    Loop While i <= RACIMOS
    ProgressBar1.Value = 0
    StatusBar1.SimpleText = "Listo"
    cmdAsignarPesos.Enabled = False
    cmdInicializarRed.Enabled = False
    cmdEntrenarRed.Enabled = True
    chkLog.Enabled = True
    chkHistorico.Enabled = True
    FILELOG = "PESOS ASIGNADOS"
    txtVisor.Text = txtVisor.Text & FILELOG & CrLf$ & _
    "-----------------------------------------------------" & CrLf$
End Sub
```

```
Public Sub cmdEntrenarRed_Click()
'Entrena la red con el archivo seleccionado y los parametros cargados
'en la frmIniciarRed
'*************************************************************************
    Dim prt, prtf, i, k, t As Integer
    Dim ApExcel As Excel.Application
    Dim ApBook As Excel.Workbook
    Dim ApSheet As Excel.Worksheet
    Set ApExcel = CreateObject("Excel.Application")
    Set ApBook = ApExcel.Workbooks.Open("c:\fpsys\archivos\pesos.xls")
    Set ApSheet = ApBook.Worksheets(1)
    'Hace que Excel no se vea
    ApExcel.Visible = False
    'Abre archivos de log para almacenar históricos y eventos
    Open "c:\fpsys\logs\log.txt" For Output Access Write As #1
    Open "c:\fpsys\logs\logfull.txt" For Output Access Write As #2
    StatusBar1.SimpleText = "Entrenando la red..."
    ProgressBar1.Max = Iteraciones - 1
    ProgressBar1.Min = 0
    ProgressBar1.Value = 0
    prt = 1
    CrLf$ = Chr(13) & Chr(10)
    FILELOG = "Entrenando la red..."
    txtVisor.Text = txtVisor.Text & FILELOG & CrLf$
    txtVisor.Refresh
    prt = 1
    'Procesa de acuerda al número de iteraciones solicitadas
    For t = 1 To Iteraciones
        CrLf$ = Chr(13) & Chr(10)
        'Guarda valores de actualizaciones de pesos en archivo histórico
        If chkHistorico.Value = 1 Then
            Print #2, Chr(10) & Chr$(13)
            Print #2, "Iteración No."; t; "-----------------------------------"
            prtf = 1
            For prtf = 1 To RACIMOS
                Print #2, "W"; prtf; "(" & PESO(prtf).PesoCee; _
                ","; PESO(prtf).PesoCol; ","; PESO(prtf).PesoDen; _
                ","; PESO(prtf).Pesolpc; ","; PESO(prtf).PesoUbi; ")"
            Next prtf
        End If
        ApSheet.Cells(t, 1) = PESO(1).PesoCol
        ApSheet.Cells(t, 2) = PESO(1).PesoUbi
        ApSheet.Cells(t, 3) = PESO(1).PesoDen
        ApSheet.Cells(t, 4) = PESO(1).Pesolpc
        ApSheet.Cells(t, 5) = PESO(1).PesoCee
        'Gurarda valor de actuahzaciones de un peso ejemplo en archivo de log
        If chkLog.Value = 1 Then
```

```
            If prt = t Then
               Print #1, "Iterac."; t; " W"; "1"; "(" & PESO(1).PesoCee; _
               ","; PESO(1).PesoCol; ","; PESO(1).PesoDen; _
               ","; PESO(1).Pesolpc; ","; PESO(1).PesoUbi; ")"
               prt = prt + 5
            End If
         End If
            For k = 1 To Numero_Datos
            Call Borrar_Indicadores.Borrar_Indicadores(FACTORP(k).COL,
FACTORP(k).UBI, _
            FACTORP(k).DEN, FACTORP(k).IPC, FACTORP(k).CEE, t)
            For i = 1 To RACIMOS
               'Calcula distancia
               Call Calcular_Distancia_Eucl.Calcular_Distancia_Eucl(i, k)
               'Calcula ganador por competencia
               If i = 1 Then
                  Ganador = i
                  Minima_distancia = DISTANCIA
               Else
                  If DISTANCIA < Minima_distancia Then
                     Ganador = i
                     Minima_distancia = DISTANCIA
                  End If
               End If
            Next i
            'Actualiza los pesos del ganador
            Call Actualizar_Pesos.Actualizar_Pesos(t, k)
         Next k
         ProgressBar1.Value = t - 1
      Next t
      'Clasifica los ganadores en un arreglo final CLASE
      ReDim CLASE(RACIMOS)
      For i = 1 To RACIMOS
         Call Actualizar_Ganador.Actualizar_Ganador(i)
      Next i
      StatusBar1.SimpleText = "Listo"
      cmdEntrenarRed.Enabled = False
      cmdProbarRed.Enabled = True
      cmdProbarArchivo.Enabled = False
      cmdEstadistica.Enabled = True
      cmdVerLog.Enabled = True
      cmdHistorico.Enabled = True
      cmdExaminar2.Enabled = True
      FILELOG = "RED ENTRENADA"
      ProgressBar1.Value = 0
      StatusBar1.SimpleText = "Listo"
      Close #1
      Close #2
```

```
    txtVisor.Text = txtVisor.Text & FILELOG & CrLf$ & _
    "-----------------------------------------------------" & CrLf$
    ApExcel.ActiveWorkbook.Save
    ApExcel.Workbooks.Close
    ApExcel.Quit
    Set ApExcel = Nothing
End Sub


Private Sub cmdEstadistica_Click()
'Muestra forma frmEstadistica
'*************************************************************************
    frmEstadistica.Show
End Sub


Private Sub cmdExaminar2_Click()
'*************************************************************************
    cdiag2.ShowOpen
    txtProbar.Text = cdiag2.FileName
    cmdProbarArchivo.Enabled = True
End Sub


Private Sub cmdFinalizarRed_Click()
'*************************************************************************
    Dim i, j As Integer
    For i = 1 To RACIMOS
        PESO(i).Centro = nil
        PESO(i).PesoCee = nil
        PESO(i).PesoCol = nil
        PESO(i).PesoDen = nil
        PESO(i).PesoIpc = nil
        PESO(i).PesoUbi = nil
        CLASE(i).PesoCee = nil
        CLASE(i).PesoCol = nil
        CLASE(i).PesoUbi = nil
        CLASE(i).PesoDen = nil
        CLASE(i).PesoIpc = nil
        CLASE(i).Centroide = nil
    Next i
```

```
    For j = 1 To Numero_Datos
        FACTORP(j).CEE = nil
        FACTORP(j).COL = nil
        FACTORP(j).DEN = nil
        FACTORP(j).IPC = nil
        FACTORP(j).UBI = nil
        FACTORP(j).Total = nil
    Next j
    StatusBar1.SimpleText = "Red Finalizada"
End Sub



Private Sub cmdHistorico_Click()
'**************************************************************************
    Dim retval
    retval = Shell("write C:\FPSys\Logs\logfull.txt", vbNormalFocus)
End Sub



Private Sub cmdInicializarRed_Click()
'Inicializa la red, creando el arreglo que contiene los datos a entrenar
'**************************************************************************
On Error GoTo fallo
    StatusBar1.SimpleText = "Inicializando la red..."
    Dim i, j, UltimoDato As Integer
    Dim ApExcel As Excel.Application
    CrLf$ = Chr(13) & Chr(10)
    'Objeto tipo Excel
    Set ApExcel = CreateObject("Excel.Application")
    FILELOG = "Iniciando la red..."
    Path_Archivo_Entrenamiento = txtPath.Text
    txtVisor.Text = txtVisor.Text & FILELOG & CrLf$
    ApExcel.Workbooks.Open (Path_Archivo_Entrenamiento)
    ApExcel.Visible = False
    i = 2
    'Calcula la cantidad de datos a entrenar
    Do While ApExcel.Cells(i, 1) <> ""
        UltimoDato = i
        i = i + 1
    Loop
    ProgressBar1.Max = UltimoDato
    ProgressBar1.Min = 0
    'Redimensiona el arreglo FACTORP con la cantidad de datos a entrenar
    ReDim FACTORP(UltimoDato - 1)
    i = 1
```

```
    j = 2
    'Carga valores del archivo Excel al arreglo FACTORP
    Do
       FACTORP(i).COL = ApExcel.Cells(j, 1)
       COL.Caption = FACTORP(i).COL
       FACTORP(i).UBI = ApExcel.Cells(j, 2)
       UBI.Caption = FACTORP(i).UBI
       FACTORP(i).DEN = ApExcel.Cells(j, 3)
       DEN.Caption = FACTORP(Í).DEN
       FACTORP(i).IPC = ApExcel.Cells(j, 4)
       IPC.Caption = FACTORP(i).IPC
       FACTORP(i).CEE = ApExcel.Cells(j, 5)
       CEE.Caption = FACTORP(i).CEE
       FACTORP(i).Total = FACTORP(i).CEE + FACTORP(i).IPC + FACTORP(i).DEN _
       + FACTORP(i).UBI + FACTORP(i).COL
       COL.Refresh
       UBI.Refresh
       DEN.Refresh
       IPC.Refresh
       CEE.Refresh
       i = i + 1
       ProgressBar1.Value = j
       j = j + 1
    Loop While i < UltimoDato
    Numero_Datos = UltimoDato - 1
    'Cierra el archivo de Excel
    ApExcel.ActiveWorkbook.Save
    ApExcel.Workbooks.Close
    ApExcel.Quit
    Set ApExcel = Nothing
    cmdInicializarRed.Enabled = False
    cmdAsignarPesos.Enabled = True
    cmdExaminar.Enabled = False
    cmdFinalizarRed.Enabled = True
    'Borra valores de indicadores
    Call Borrar_Indicadores.Borrar_Indicadores(0, 0, 0, 0, 0, 0)
    FILELOG = "RED INICIALIZADA"
    txtVisor.Text = txtVisor.Text & FILELOG & CrLf$ & _
    "-------------------------------------------------------" & CrLf$
    ProgressBar1.Value = 0
    StatusBar1.SimpleText = "Listo"
    'Llama a forma para ingresar valores de entrenamiento
    frmIniciarRed.Show
    Exit Sub
fallo:
    MsgBox "Archivo no válido o No ha ingresado el nombre dei Arhivo de
Entrenamiento"
End Sub
```

```
Private Sub cmdProbarArchivo_Click()
'Prueba archivo de datos en red entrenada
'*************************************************************************
On Error GoTo fallo
   Path_Archivo_Resultados = txtProbar.Text
   CrLf$ = Chr(13) & Chr(10)
   'Llama a procedimiento para probar datos en la red
   Call Probar_Archivo.Probar_Archivo(Path_Archivo_Resultados)
   FILELOG = "Prueba Finalizada... "
   txtVisor.Text = txtVisor.Text & FILELOG & CrLf$ & _
   "-------------------------------------------" & CrLf$
   frmProbarArchivo.lblProbar.Caption = "Se han ingresado " _
   & Numero_Datos & " datos del archivo " & Path_Archivo_Resultados & _
   ". Los resultados generados por la red se han almacenado en el mismo archivo."
   frmProbarArchivo.Show
   frmMain.ProgressBar1.Value = 0
   Exit Sub
fallo:
   MsgBox "Archivo no válido o No ha ingresado el nombre del Arhivo"
End Sub



Private Sub cmdProbarRed_Click()
'Muestra forma frmProbarRed
'*************************************************************************
   frmProbarRed.Show
End Sub



Private Sub cmdSalir_Click()
'*************************************************************************
   End
End Sub


Private Sub cmdVerLog_Click()
'Abre archivo de log en textbox
'*************************************************************************
   StatusBar1.SimpleText = "Abriendo Archivo..."
   On Error GoTo fallo
   X = GetAttr("c:\fpsys\logs\log.txt")
   CrLf$ = Chr(13) & Chr(10)
   Archivo = "c:\fpsys\logs\log.txt"
   txtVisor.Text = txtVisor.Text & "ARCHIVO DE LOG" & CrLf$
   txtVisor.Text = txtVisor.Text & "-------------------------------------" & CrLf$
   Open Archivo For Input Access Read As #1
```

```
    While Not EOF(1)
        Line Input #1, file_data$
        txtVisor.Text = txtVisor.Text & file_data$ & CrLf$
    Wend
    Close #1
    StatusBar1.SimpleText = "Listo"
    Exit Sub
fallo:
    MsgBox "El archivo no existe."
    StatusBar1.SimpleText = "Listo"
End Sub



Private Sub Form_Load()
'************************************************************************
    cmdExaminar.Enabled = True
    cmdExaminar2.Enabled = False
    cmdInicializarRed.Enabled = False
    cmdProbarRed.Enabled = False
    cmdVerLog.Enabled = False
    cmdHistorico.Enabled = False
    cmdProbarArchivo.Enabled = False
    cmdEstadistica.Enabled = False
    cmdFinalizarRed.Enabled = False
    cmdEntrenarRed.Enabled = False
    cmdAsignarPesos.Enabled = False
End Sub



Private Sub cmbCee_Click()
'************************************************************************
    If cmbCee.Text = "Hasta 100.000" Then lblCee.Caption = "1"
    If cmbCee.Text = "Mas de 100.000 Hasta 200.000" Then lblCee.Caption = "2"
    If cmbCee.Text = "Mas de 200.000 Hasta 300.000" Then lblCee.Caption = "3"
    If cmbCee.Text = "Mas de 300.000 Hasta 500.000" Then lblCee.Caption = "5"
    If cmbCee.Text = "Mas de 500.000" Then lblCee.Caption = "6"
    lblCee.Refresh
End Sub



Private Sub cmbCol_Click()
'************************************************************************
    If cmbCol.Text = "Fiscal" Then lblcol.Caption = "1"
    If cmbCol.Text = "Particular menos de 100.000" Then _
    lblcol.Caption = "1"
```

```
    If cmbCol.Text = "Particular mas de 100.000 hasta 200.000" Then _
    lblcol.Caption = "2"
    If cmbCol.Text = "Particular mas de 200.000 hasta 300.000" Then _
    lblcol.Caption = "3"
    If cmbCol.Text = "Particular mas de 300.000 hasta 400.000" Then _
    lblcol.Caption = "4"
    If cmbCol.Text = "Particular mas de 400.000 hasta 500.000" Then _
    lblcol.Caption = "5"
    If cmbCol.Text = "Particular mas de 500.000 hasta 600.000" Then _
    lblcol.Caption = "6"
    If cmbCol.Text = "Particular mas de 600.000 hasta 700.000" Then _
    lblcol.Caption = "7"
    If cmbCol.Text = "Particular mas de 700.000 hasta 800.000" Then _
    lblcol.Caption = "8"
    If cmbCol.Text = "Particular mas de 800.000 hasta 900.000" Then _
    lblcol.Caption = "9"
    If cmbCol.Text = "Particular mas de 900.000 hasta 1.200.000" Then _
    lblcol.Caption = "10"
    If cmbCol.Text = "Particular mas de 1.200.000 hasta 1.500.000" Then _
    lblcol.Caption = "11"
    If cmbCol.Text = "Particular mas de 1.500.000" Then _
    lblcol.Caption = "12"
    lblcol.Refresh
End Sub



Private Sub cmbDen_Click()
'*********************************************************************
    If cmbDen.Text = "Menos de 18m2" Then lblDen.Caption = "1"
    If cmbDen.Text = "Mas de 18m2 y Menos de 36m2" Then lblDen.Caption = "2"
    If cmbDen.Text = "Mas de 36m2 y Menos de 54m2" Then lblDen.Caption = "3"
    If cmbDen.Text = "Mas de 54m2" Then lblDen.Caption = "4"
    lblDen.Refresh
End Sub

Private Sub cmbIpc_Click()
'*********************************************************************
    If cmbIpc.Text = "Mas de 100.000 Hasta 400.000" Then lblIpc.Caption = "0"
    If cmbIpc.Text = "Mas de 400.000 Hasta 1.000.000" Then lblIpc.Caption = "2"
    If cmbIpc.Text = "Mas de 1.000.000 Hasta 2.000.000" Then lblIpc.Caption = "3"
    If cmbIpc.Text = "Mas de 2.000.000" Then lblIpc.Caption = "4"
    lblIpc.Refresh
End Sub
```

```
Private Sub cmbUbi_Click()
'*************************************************************************
    If cmbUbi.Text = "Guayaquil en Sector Popular" Then lblUbi.Caption = "0"
    If cmbUbi.Text = "Guayaquil en Sector Residencial Social" Then _
    lblUbi.Caption = "2"
    If cmbUbi.Text = "Guayaquil en Sector Residencial Medio" Then _
    lblUbi.Caption = "6"
    If cmbUbi.Text = "Guayaquil en Sector Residencial Medio Alto" Then _
    lblUbi.Caption = "10"
    If cmbUbi.Text = "Guayaquil en Sector Residencial Alto" Then _
    lblUbi.Caption = "12"
    If cmbUbi.Text = "Guayaquil en Sector Residencial Exclusivo" Then _
    lblUbi.Caption = "14"
    If cmbUbi.Text = "Quito en Sector Residencial Social" Then _
    lblUbi.Caption = "0"
    If cmbUbi.Text = "Quito en Sector Residencial Medio" Then _
    lblUbi.Caption = "6"
    If cmbUbi.Text = "Quito en Sector Residencial Alto" Then _
    lblUbi.Caption = "12"
    If cmbUbi.Text = "Galapagos" Then lblUbi.Caption = "2"
    If cmbUbi.Text = "Otra Provincia en Sector Popular" Then _
    lblUbi.Caption = "0"
    If cmbUbi.Text = "Otra Provincia en Sector Residencial Medio" Then _
    lblUbi.Caption = "6"
    lblUbi.Refresh
End Sub



Private Sub cmdAceptar_Click()
'*************************************************************************
    Dim i, j, k As Integer
    Dim dato_procesado As String
    ReDim FACTORP(1)
    i = 1
    'Carga valores de datos del archivo de Excel al Arreglo FACTORP
        FACTORP(i).COL = Int(lblcol.Caption)
        FACTORP(i).UBI = lblUbi.Caption
        FACTORP(i).DEN = lblDen.Caption
        FACTORP(i).IPC = lblIpc.Caption
        FACTORP(i).CEE = lblCee.Caption
        FACTORP(i).Total = 0
    i = 1
    k = 1
    'Prueba los datos en la red
    For i = 1 To RACIMOS
        'Calcula distancia
        DISTANCIA = 0
```

```
        DISTANCIA = DISTANCIA + (Abs(CLASE(i).PesoCee - FACTORP(k).CEE)) ^ 2
        DISTANCIA = DISTANCIA + (Abs(CLASE(i).PesoCol - FACTORP(k).COL)) ^ 2
        DISTANCIA = DISTANCIA + (Abs(CLASE(i).PesoDen - FACTORP(k).DEN)) ^ 2
        DISTANCIA = DISTANCIA + (Abs(CLASE(i).PesoIpc - FACTORP(k).IPC)) ^ 2
        DISTANCIA = DISTANCIA + (Abs(CLASE(i).PesoUbi - FACTORP(k).UBI)) ^ 2
        'Calcula ganador por competencia
        If i = l Then
           Ganador = i
           Minima_distancia = DISTANCIA
        Else
           If DISTANCIA < Minima_distancia Then
              Ganador = i
              Minima_distancia = DISTANCIA
           End If
        End If
     Next i
     i = 0
     i = Int(lblcol.Caption) + Int(lblIpc.Caption) _
     + Int(lblDen.Caption) + Int(lblUbi.Caption) + Int(lblCee.Caption)

     CLASE(Ganador).Centroide = CLASE(Ganador).PesoCee + CLASE(Ganador).PesoIpc
_
     + CLASE(Ganador).PesoDen + CLASE(Ganador).PesoUbi + CLASE(Ganador).PesoCol
     lblfactorp.Caption = CLASE(Ganador).Centroide
     lblfactor.Caption = i
     lblfactor.Refresh
     lblfactorp.Refresh
     cmdAceptar.Enabled = False
End Sub



Private Sub cmdSalir_Click()
'************************************************************************
     Unload frmProbarRed
End Sub



Private Sub Form_Load()
'************************************************************************
     cmbCol.Clear
     cmbIpc.Clear
     cmbDen.Clear
     cmbUbi.Clear
     cmbCee.Clear
     cmbCol.AddItem ("Fiscal")
     cmbCol.AddItem ("Particular menos de 100.000")
```

```
    cmbCol.AddItem ("Particular mas de 100.000 hasta 200.000")
    cmbCol.AddItem ("Particular mas de 200.000 hasta 300.000")
    cmbCol.AddItem ("Particular mas de 300.000 hasta 400.000")
    cmbCol.AddItem ("Particular mas de 400.000 hasta 500.000")
    cmbCol.AddItem ("Particular mas de 500.000 hasta 600.000")
    cmbCol.AddItem ("Particular mas de 600.000 hasta 700.000")
    cmbCol.AddItem ("Particular mas de 700.000 hasta 800.000")
    cmbCol.AddItem ("Particular mas de 800.000 hasta 900.000")
    cmbCol.AddItem ("Particular mas de 900.000 hasta 1.200.000")
    cmbCol.AddItem ("Particular mas de 1.200.000 hasta 1.500.000")
    cmbCol.AddItem ("Particular mas de 1.500.000")
    cmbUbi.AddItem ("Guayaquil en Sector Popular")
    cmbUbi.AddItem ("Guayaquil en Sector Residencial Social")
    cmbUbi.AddItem ("Guayaquil en Sector Residencial Medio")
    cmbUbi.AddItem ("Guayaquil en Sector Residencial Medio Alto")
    cmbUbi.AddItem ("Guayaquil en Sector Residencial Alto")
    cmbUbi.AddItem ("Guayaquil en Sector Residencial Exclusivo")
    cmbUbi.AddItem ("Quito en Sector Residencial Social")
    cmbUbi.AddItem ("Quito en Sector Residencial Medio")
    cmbUbi.AddItem ("Quito en Sector Residencial Alto")
    cmbUbi.AddItem ("Galapagos")
    cmbUbi.AddItem ("Otra Provincia en Sector Popular")
    cmbUbi.AddItem ("Otra Provincia en Sector Residencial Medio")
    cmbDen.AddItem ("Menos de 18m2")
    cmbDen.AddItem ("Mas de 18m2 y Menos de 36m2")
    cmbDen.AddItem ("Mas de 36m2 y Menos de 54m2")
    cmbDen.AddItem ("Mas de 54m2")
    cmbCee.AddItem ("Hasta 100.000")
    cmbCee.AddItem ("Mas de 100.000 Hasta 200.000")
    cmbCee.AddItem ("Mas de 200.000 Hasta 300.000")
    cmbCee.AddItem ("Mas de 300.000 Hasta 500.000")
    cmbCee.AddItem ("Mas de 500.000")
    cmbIpc.AddItem ("Mas de 100.000 Hasta 400.000")
    cmbIpc.AddItem ("Mas de 400.000 Hasta 1.000.000")
    cmbIpc.AddItem ("Mas de 1.000.000 Hasta 2.000.000")
    cmbIpc.AddItem ("Mas de 2.000.000")
End Sub



Private Sub cmdFactor_Click()
'Muestra Gráfico correspondiente al resultado de los racimos
'generados en una prueba de datos
'*********************************************************************
    StatusBar1.SimpleText = "Generando Clasificación..."
    'Llama a modulo que ordena los datos y/os dasifica
    Call Generar_Tabla_Resultados.Generar_Tabla_Resultados
    Dim i As Integer
```

```
    Dim arrValores()
    ReDim arrValores(RACIMOS, 2)
    Dim ApExcel As Excel.Application
    Dim ApBook As Excel.Workbook
    Dim ApSheet As Excel.Worksheet
    Set ApExcel = CreateObject("Excel.Application")
    Set ApBook = ApExcel.Workbooks.Open(frmMain.txtProbar)
    Set ApSheet = ApBook.Worksheets(2)
    ApExcel.Visible = False
    MSChart1.TitleText = "Clasificación Obtenida"
    MSChart1.ChartType = VtChChartType2dBar
    ProgressBar1.Value = 0
    ProgressBar1.Max = RACIMOS
    StatusBar1.SimpleText = "Generando Gráfico..."
    'Cargo datos ordenados y dasificados en arreglo
    For i = 1 To RACIMOS
        ApSheet.Cells(i, 1) = i
        arrValores(i, 2) = ApSheet.Cells(i, 2)
        arrValores(i, 1) = "C" & i
        frmEstadistica.ProgressBar1.Value = i
    Next i
    'Genero gráfico con los datos del arreglo generado
    MSChart1.ChartData = arrValores
    ApExcel.ActiveWorkbook.Save
    ApExcel.Workbooks.Close
    ApExcel.Quit
    Set ApExcel = Nothing
    StatusBar1.SimpleText = "Listo"
End Sub



Private Sub cmdPesos_Click()
'Muestra Gráfico correspondiente al log
'sobre desarrollo y actualización de pesos
*************************************************************************
    Dim i As Integer
    Dim arrValores()
    ReDim arrValores(25, 5)
    StatusBar1.SimpleText = "Recuperando histórico..."
    ProgressBar1.Value = 0
    ProgressBar1.Max = 25
    MSChart1.TitleText = "Evolucion de Pesos"
    'Acceso a archivo Pesos.xls
    Dim ApExcel As Excel.Application
    Set ApExcel = CreateObject("Excel.Application")
    ApExcel.Workbooks.Open ("c:\fpsys\archivos\pesos.xls")
    ApExcel.Visible = False
```

```
    MSChart1.ChartType = VtChChartType3dLine
    StatusBar1.SimpleText = "Generando Gráfico..."
    'Genera Arreglo con datos de los pesos
    For i = 1 To 25
        arrValores(i, 1) = ApExcel.Cells(i, 1)
        arrValores(i, 2) = ApExcel.Cells(i, 2)
        arrValores(i, 3) = ApExcel.Cells(i, 3)
        arrValores(i, 4) = ApExcel.Cells(i, 4)
        arrValores(i, 5) = ApExcel.Cells(i, 5)
        ProgressBar1.Value = i
    Next i
    'Genera gráfico a través del arreglo anterior
    MSChart1.ChartData = arrValores
    ApExcel.Workbooks.Close
    ApExcel.Quit
    Set ApExcel = Nothing
    StatusBar1.SimpleText = "Listo"
End Sub



Private Sub cmdSalir_Click()
'**********************************************************************
    Unload Me
End Sub
```

CAPÍTULO 6

6. ANALÍSIS DE RESULTADOS

6.1. TABULACIÓN DE RESULTADOS, ESTADÍSTICAS Y ANALISIS

Una de las metas propuestas en este proyecto es proveer a la Escuela Superior Politécnica del Litoral de una herramienta que le permita manejar estadísticas y crear diversos escenarios permitiendo así, manipular los datos y valorizar los criterios que determinan el factor P.

El propósito fundamental de esta tesis es valorar la ponderación asignada por el factor P, por lo cual el primer reto representa realizar una comparación entre el valor P asignado por la fórmula original y el valor P asignado por la red neuronal. Este objetivo debe ser validado inicialmente para un estudiante.

Hemos escogido un estudiante promedio con las siguientes características:

COL: 9 (Colegio Particular Y pensión 840.000)

UBI: 6 (Sector Residencial Medio)

DEN: 3 (Densidad = 150 mts^2/4 = 37.5 mts^2)

IPC: 3 (Ingreso = (12'OOO.OOO - 4'000.000)/4 = 2'OOO.OOO)

CEE: 5 (Promedio mensual de consumo eléctrico = 450.000)

FACTOR P = COL + UBI + DEN + IPC + CEE

Valor asignado por la fórmula	26
Valor asignado por la red neuronal	26

Para este caso el estudiante obtiene los siguientes resultados para el Valor del Factor P, tanto con la fórmula original y la red neuronal:

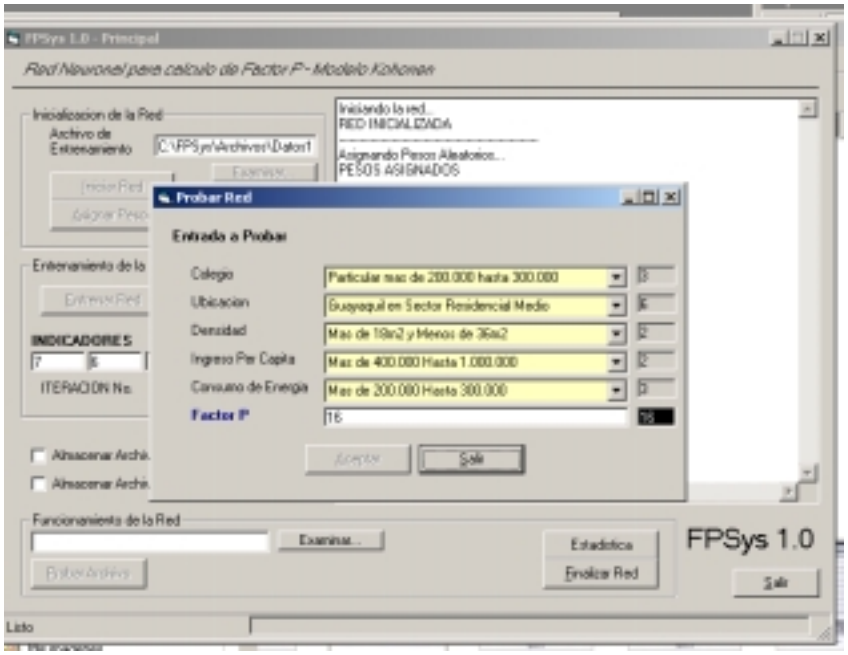


Figura 6.1.1

Este resultado demuestra que la red neuronal logra aprender el funcionamiento de la asignación del valor P, y puede generar los mismos

valores que la fórmula a través del aprendizaje basado en una red neuronal. Este tipo de auto - aprendizaje brinda la ventaja de mantenerse constante, de tal forma que las variaciones de los datos en la comunidad o universo, van a ser reflejados en los valores asignados, tarea que no puede ser realizada con una fórmula rígida.

Adicionalmente, en el momento de asignar un nuevo valor a un estudiante, la base de datos ingresa un nuevo elemento al sistema de aprendizaje, lo cual permite fortificar los patrones ya existentes o diferencia patrones que aún no habían sido suficientemente dispersos.

Nuestro siguiente reto u objetivo, en el desarrollo de esta tesis es valorar los resultados obtenidos por más de un elemento o estudiante simultáneamente.

Para este objetivo es válido establecer un grupo de estudiantes con diversas características, y analizar el resultado obtenido para el Factor P en la red neuronal versus el resultado obtenido con la fórmula original.

Hemos realizado este ejercicio con un grupo de estudiantes (30), a los cuales se les ha asignado su factor P a través de la fórmula original,

como también han sido ingresados a la red neuronal para obtener los siguientes resultados:

COL	UBI	DEN	IPC	CEE	Fórmula	Red Neuronal
3	0	1	2	1	7	8
3	2	1	0	1	7	8
2	2	1	3	2	10	10
5	0	1	2	2	10	10
1	6	2	3	3	15	14
3	6	1	3	2	15	15
6	2	1	4	2	15	16
4	2	1	3	5	15	15
4	6	3	3	3	19	19
4	6	2	4	3	19	19
4	6	2	4	3	19	19
5	10	2	3	3	23	20
8	6	3	3	3	23	21
10	6	1	3	3	23	23
9	4	4	3	3	23	25
5	10	3	4	5	27	27
6	10	3	3	5	27	28
11	6	2	3	5	27	28
7	10	3	3	5	28	*28*
12	10	2	3	3	30	28
6	12	4	4	5	31	31
9	10	3	4	5	31	32
10	12	3	3	3	31	29
7	12	4	4	5	32	28
11	10	3	3	5	32	32
9	12	3	4	5	33	33
9	12	3	4	5	33	36
11	10	3	3	6	33	32
12	12	4	4	5	37	36
10	14	4	4	6	38	36

Tabla 6.1.1

Estos resultados pueden ser mejor analizados utilizando diagramas estadísticos que permitan visualizar fácilmente las variaciones y las asignaciones otorgadas:

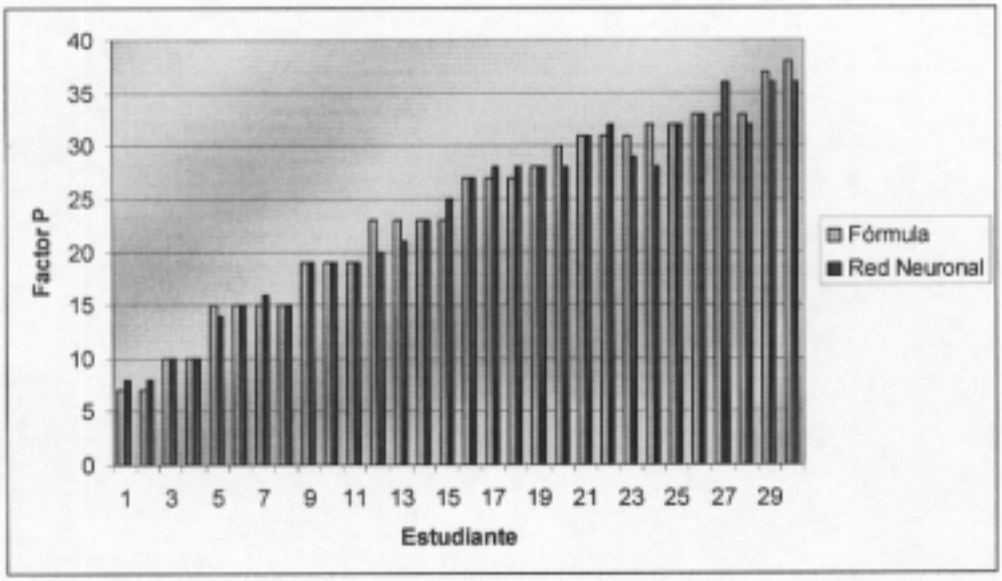


Figura 6.1.2

Como puede notarse, los resultados obtenidos en la red neuronal son muy semejantes a los generados por la fórmula del Factor P. Estos resultados fueron generados a través de 100 iteraciones, con lo cual queremos notar que el grado de exactitud de los resultados son directamente proporcional al número de iteraciones por las cuales, los datos sean tratados.

En este caso, hemos tenido que un 49 % de los estudiantes han tomado el mismo factor P calculado con la fórmula original, un 33 % ha obtenido un factor P que difiere en una unidad con respecto al factor P de la fórmula, 16 % ha obtenido un factor P que difiere en 2 unidades al valor de la fórmula, y solamente, 2% obtuvo un factor P que difieren en más de 3 unidades.

Estos valores pueden ser optimizados, aumentado el número de iteraciones, con lo cual se mejoran los patrones, pero a su vez, se requiere más tiempo para el entrenamiento de la red y la obtención de los resultados.

Creemos que estos valores, son muy satisfactorios y que cumplen con el objetivo de la red y la tesis propuesta.

La tercera funcionalidad que queremos obtener de la red neuronal, es la capacidad de poder manejar nuevos escenarios que nos permitan evaluar cambios en las asignaciones y el impacto obtenido en dichos cambios.

Adicionalmente, se desea presentar diferentes modelos en donde podamos variar el rango del factor P para una determinada cantidad de

datos y verificar el comportamiento de los resultados en función de esta variable.

Como se describió en el primer capítulo de esta tesis, el factor P varía en un rango que va de 3 a 40, esto es, existen 38 valores posibles para ponderar el factor.

Escenario	Rango	Valores
1	3—40	38

Tabla 6.1.2

Para proveer de los mencionados escenarios a la herramienta, se ha variado este rango de la siguiente manera:

Escenario	Rango	Cantidad de valores
1	**3—40**	**38**
2	**3—50**	**48**
3	**3—30**	**28**

Tabla 6.1.3

Luego de procesar los datos en FPSys 1.0 se obtuvieron diversos resultados basándose en la cantidad de valores del Factor P solicitado.

Esta información nos resulta muy valiosa para hacer un estudio específico de los valores obtenidos.

Es importante mencionar que los datos utilizados para entrenar la red han sido obtenidos gracias a la colaboración de la Unidad del CRECE de la Universidad, los cuales corresponden a junio de 1999. Adicionalmente, se han utilizado datos obtenidos a través de un muestreo en las diferentes facultades de la Universidad.

Con el propósito de conocer los resultados a continuación se presenta una tabla que contiene los resultados obtenidos en cada uno de los escenarios y sus conclusiones más valiosas.

COL	UBI	DEN	IPC	CEE	SUMA
2	2	1	3	3	11
1	6	2	3	3	15
3	0	1	2	1	7
4	0	3	2	2	11
2	2	3	3	3	13
2	2	3	3	3	13
3	2	1	0	1	7
3	6	1	3	2	15
2	2	1	3	2	10
3	2	2	3	3	13
3	2	2	3	3	13
4	6	3	3	3	19
5	10	3	3	5	26
4	4	1	2	2	13
4	2	1	3	2	12
4	6	2	4	3	19
4	2	1	3	2	12
6	2	1	4	2	15
4	6	1	2	5	18
4	2	1	3	5	15
4	2	1	3	6	16
4	6	2	4	3	19

4	10	3	3	5	25
5	0	1	2	2	10
5	2	1	3	2	13
5	6	2	3	3	19
5	10	2	3	3	23
5	6	2	4	5	22
5	10	3	4	5	27
6	6	2	2	2	18
6	10	3	3	3	25
6	10	3	3	5	27
6	12	4	4	5	31
7	6	2	2	3	20
7	6	2	2	3	20
7	6	3	3	5	24
7	10	3	3	5	28
7	12	4	4	5	32
8	6	1	2	2	19
8	6	2	3	3	22
8	6	3	3	3	23
8	6	3	3	5	25
9	6	2	4	6	27
9	10	2	3	3	27
9	10	3	4	5	31
9	12	3	4	5	33
9	12	3	4	5	33
10	6	1	3	3	23
10	10	2	3	3	28
10	12	3	3	3	31
10	14	4	4	6	38
11	6	2	3	3	25
11	10	3	3	6	33
11	6	2	3	5	27
11	10	3	3	5	32
11	6	4	4	3	28
12	10	2	3	3	30
12	12	4	4	5	37
9	4	4	3	3	23

Tabla 6.1.4

Estos son los valores que se obtienen directamente de sumar los datos de los estudiantes, utilizando la fórmula original del Factor P.

En la figura 6.1.3 se muestra los factores P obtenidos por estos datos utilizando su fórmula original.

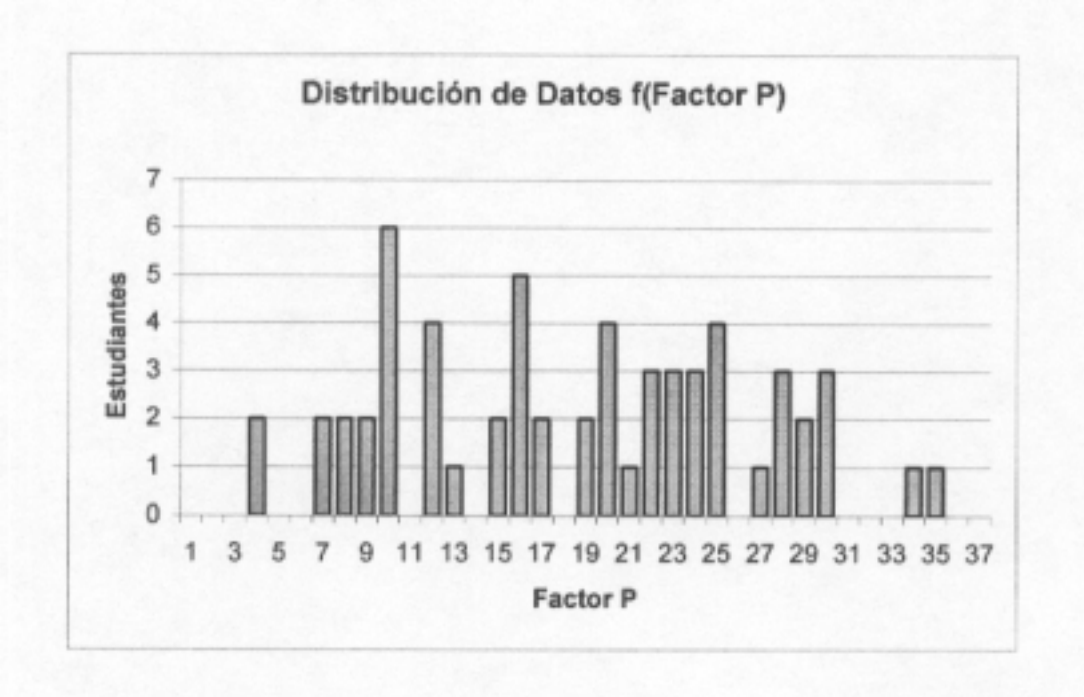


Tabla 6.1.3

Una vez obtenidos estos datos, pueden ser ingresados a la red entrenada para que el sistema otorgue el Factor P a los estudiantes sin hacer uso de la fórmula de sumatoria. Los datos entregados por la red son los siguientes:

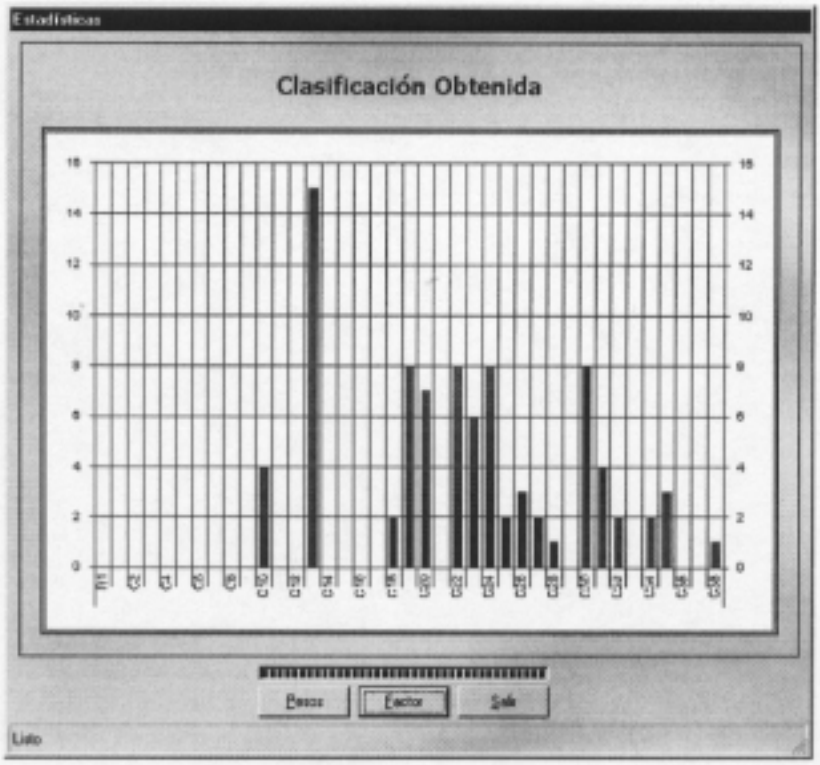


Figura 6.1.4

De la comparación de este gráfico con el gráfico original podemos notar que los valores obtenidos para el Factor P del 18 al 35 son similares, lo cual indica que el sistema ubica a los estudiantes de manera muy semejante a la fórmula original. Existe un desfase para los valores 7,10 y 11, esto como respuesta a una cantidad pequeñas de datos para estas clases de estudiantes.

En cualquiera de los escenarios planteados, se ha puesto a prueba el nivel de estabilidad en el esquema de ponderación y pesos, con la finalidad de verificar que las rutinas y procesos que permiten las

iteraciones y el autoaprendizaje, funcionen de manera rápida, eficiente y óptima.

En el siguiente gráfico se muestra cuando se estabilizan los pesos en el proceso de entrenamiento de la red en función del tiempo.

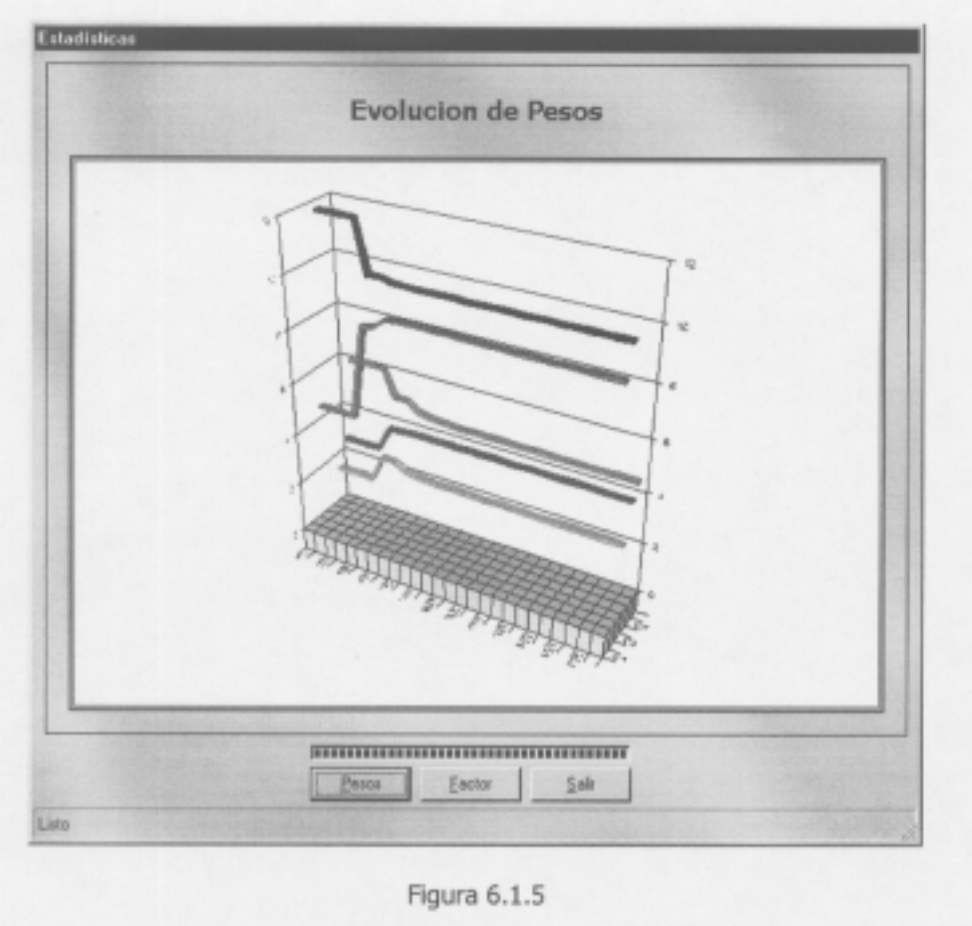


Figura 6.1.5

La red logra estabilizarse durante la iteración 40 aproximadamente, esto es, luego de la 40ava iteración la red no sufre alteración en la asignación de pesos para cada una de las clases que representa los valores P obtenidos. De este resultado podemos concluir que la red

neuronal lograr crear los vecindarios y patrones rápidamente y se mantiene estables y consistentes, lo cual permite obtener resultados confiables y veraces.

Utilizando la red neuronal para el segundo escenario, variamos la cantidad de valores posibles para el factor P, utilizando 48 valores posibles que van de 3 a 50. Como resultado obtenemos la siguiente distribución de estudiantes:

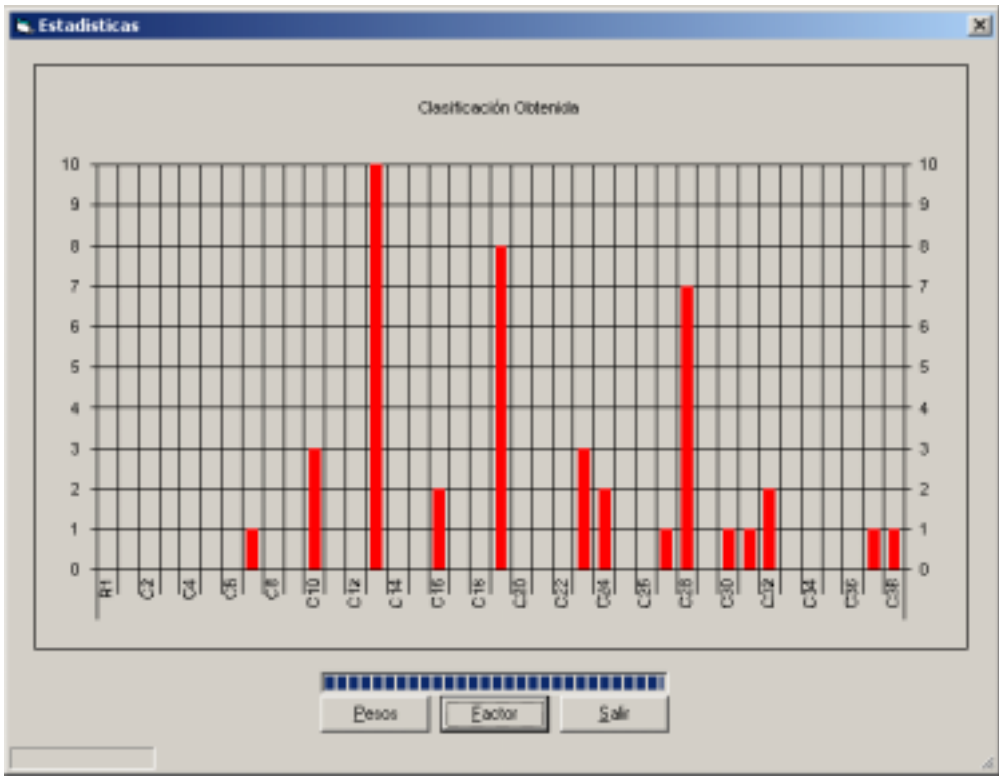


Figura 6.1.6

Dado que los valores del Factor P en el rango del 18 al 40 no varían considerablemente, se concluye que el aumentar más clases al factor no altera la asignación para los mismos datos. Se mantiene el mismo fenómeno para lo valores bajos debido a los pocos datos ingresados para esta clase a la red.

Como primera importante conclusión, se puede mencionar que el valor máximo del Factor P 40 se encuentra muy acorde con la realidad del problema y es vano incrementar este límite ya que las asignaciones causadas serían las mismas.

Utilizando la red neuronal para el tercer escenario, variamos la cantidad de valores posibles para el factor P, utilizando 28 valores posibles que van de 3 a 30. Como resultado obtenemos la siguiente distribución de estudiantes:

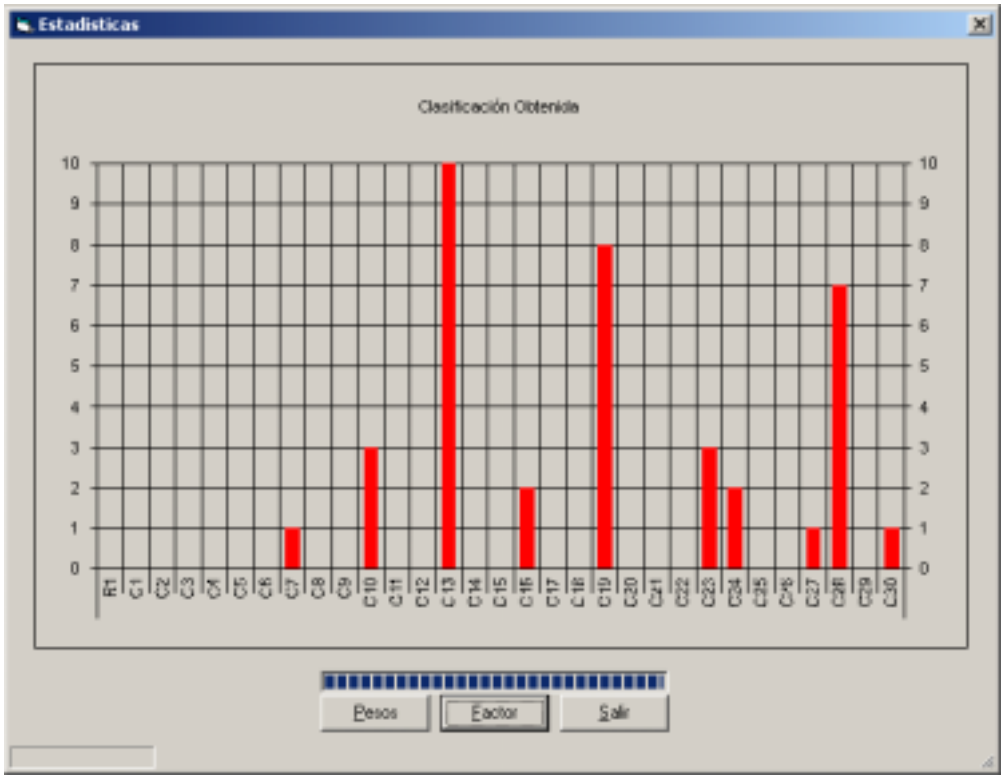


Figura 6.1.7

En este escenario ahora variamos el límite del Factor P y se coloca un valor más bajo. Como resultado podemos notar que para un modelo en el cual se clasifique a los estudiantes con un factor P con un rango del 3 al 30, la red distribuye de manera proporcional los valores asignados.

CONCLUSIONES

La primera conclusión de este proyecto, se da en la efectividad de la asignación de valores a través de una red neuronal, siendo así demostrado que existe una nueva opción para el tratamiento de datos a través de modelos humanos.

Demostramos también que el computador es capaz de reconocer patrones, lo cual puede ser utilizado para identificar elementos y clasificar datos.

El principal objetivo de esta tesis, fue evaluar la asignación del factor P para un estudiante a través del modelo neuronal, lo cual fue obtenido satisfactoriamente, para lo cual demostramos la efectividad del algoritmo y modelo de Kohonen.

Los resultados obtenidos en esta tesis permiten esquematizar varios escenarios los cuales han sido analizados en el capítulo anterior, de aquí, que el sistema permite ingresar datos y evaluar ciertas situaciones especificas.

El rango del Factor P asignado por la ESPOL, es simulado por FPsys 1.0, el cual entrega resultados muy similares a los obtenidos por la Universidad, lo cual facilitaría hacer ciertas modificaciones en los rangos y obtener resultados muy cercanos a la realidad. Este estudio permite verificar que las asignaciones hechas para el factor P por la Universidad, son reproducidas con un sistema a través de un mecanismo sistematizado, lo cual implica su correcta asignación y ponderación para cada uno de los criterios tomados para asignar el Valor del Factor P.

Basándose en los escenarios analizados en el capítulo anterior, el cambio de rango para el valor del factor P, no produciría cambios drásticos que impacten fuertemente a los costos que los estudiantes cancelan por concepto de matriculación.

Demostrando así, que existe una manera de resolver situaciones o problemáticas sin utilizar los algoritmos y métodos tradicionales. Ahora se cuenta con una nueva opción como son los sistemas basados en redes neuronales o expertos.

Como ejemplo práctico, tenemos en el área de telecomunicaciones un estudio basado en una red neuronal. Uno de los objetivos más importantes en el diseño o implementación de una red de radio es cubrir

el máximo área posible con el mínimo de equipos o transmisores, permitiendo un ahorro significativo en lo que concierne a antenas repetidoras. Se puede llegar a establecer el número mínimo de antenas utilizando un algoritmo neuronal que permita identificar una relación entre la cobertura de un dispositivo y el numero de equipos requeridos, por ejemplo:

$$f(x) = \frac{Cobertura(x)^{\alpha}}{Numerotransmisores(x)}$$

Un caso práctico real es un proyecto europeo que está siendo desarrollado por la empresa Teleasistencia Cardiotest y AICIA de la Universidad de Sevilla, representado en un dispositivo llamado **CardioSmart**. Este proyecto consiste en el diseño y la realización de un electrocardiógrafo portátil e inteligente con enlace radio, capaz de monitorizar la actividad cardiaca de los pacientes detectando automáticamente los principales fallos del corazón y transmitiéndolo posteriormente a un centro médico receptor.

El principal objetivo ha sido perfeccionar la calidad de vida de los pacientes cardiológicos para evitar que deban de ser confinados en hospitales, sintiéndose libres de caminar e incluso viajar a cualquier parte bajo la supervisión de un cardiólogo o un especialista de un hospital. La atención médica es casi instantánea gracias a un canal de voz que incluye

el enlace GSM y permitirá una conexión directa en caso de emergencia. Algunas arritmias van a ser detectadas utilizando técnicas de reconocimiento de patrones a travès de una red nueronal. Tras la comparación de la señal con el electrocardiograma normal una red neuronal es capaz de clasificar las diferentes clases de enfermedades del corazón.

Finalmente, queremos notar la facilidad del sistema para evaluar situaciones o simular escenarios. Este tipo de funcionalidad, no es incluida en un sistema de algoritmo tradicional, a diferencia, de un sistema de algoritmo basado en red neuronal, el cual puede ingresar datos y esquematizar escenarios diferentes a los reales, con el propósito de evaluar los resultados. Esta funcionalidad es muy valiosa en el caso de evaluar sistemas que necesitan utilizar límites en sus parámetros y disparar alarmas al acercarse al umbral de estos parámetros. Este tipo de sistemas, puede salvar vidas en casos de sistemas industriales o de seguridad.

BIBLIOGRAFÍA

Conceptos Fundamentales sobre las Redes Neuronales
Marcelo González, Buenos Aires, 1998

Elements of Artificial Neural Networks
Sanjeev lambe — Bhasker KulKarmi, India, 1996

Redes neurona/es. Algoritmos, aplicaciones y técnicas de programación.
James A. Freeman y David M. Skapura. Ed. Addison-Wesley/Díaz de Santos.

Introduction to artificial intelligence.
Phillip C. Jackson, Jr. Dover Publications, Inc. New York.

Seff Organization and Associative Memory
Kohonen, 1. 1984. Springer-Verlag, Berlín.

Neural Networks for Pattern Recognition
Cristopher M. Bishop, 1995

An Introduction to Neural Nets
James A. Anderson

Computer Based Training on Neural Nets
Richard Jackes, 1998

Kohonen Maps
Samuel Kaski, 1998

Computation / Neuroscience
James Bower, Jul 1996

Unsupervised Net Kohonen's Self Oganizing Feature Map
Dagmar Niebur, 1995

MANUAL DE INSTALACIÓN

Medios

Se ha generado un CDROM que contiene los instaladores del software desarrollado y adicionalmente la siguiente documentación:

Instaladores FPSys 1.0

Contiene los archivos necesarios para instalar el aplicativo FPSys 1.0.

Ayuda FPSys 1.0

Contiene los archivos de ayuda para la aplicación.

Documentación completa - Tesis

Contiene los documentos electrónicos sobre la tesis, capítulos, manuales, etc.

Instaladores de Software

Contiene los instaladores de algunos de los aplicativos que se presentaron en el Capítulo 4.

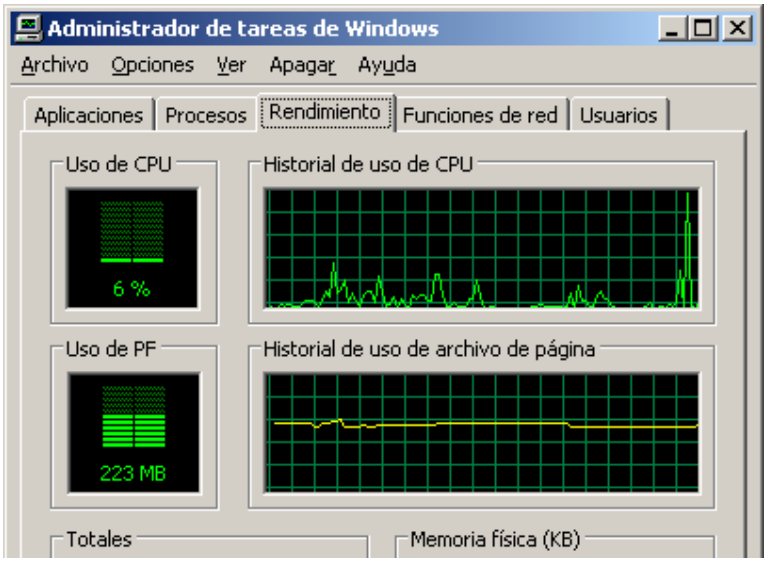
Requerimientos de Hardware

FPSys 1.0 requiere las siguientes especificaciones de Hardware y Software:

- 486 SX, 486 SL, 486 DX* o Pentium*.
- Soporte Gráfico EGA, VGA* o SVGA* (Monitor y tarjeta de video).
- 16 MB (32MB*) de memoria RAM.
- 9 MB de espacio libre en disco para aplicativo.
- 2 MB de espacio libre en disco para ayuda.
- 1 MB de espacio libre en disco para archivos de ejemplo.
- Mouse compatible PS/2
- Unidad de CD-ROM
- Windows 95, 98, NT/2000/XP
- Excel 97/2000/XP
- Mínimo 520 KB de memoria libre luego de iniciar Windows.

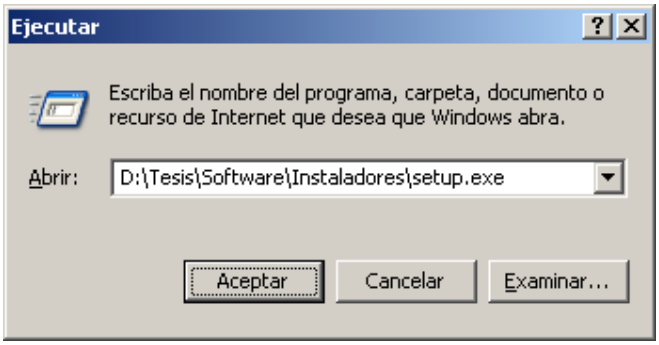
Con propósito de evaluar los consumos de recursos, los siguientes gráficos muestran el consumo de memoria y procesador durante la

ejecución del proceso de entrenamiento de la red neuronal en un equipo con procesador INTEL 1 GHZ y 128 Mbytes de memoria RAM:



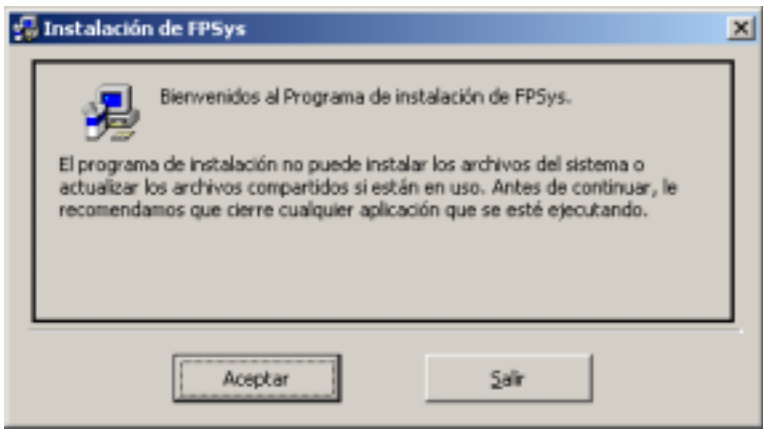
Instalación

1. Desde **Ejecutar(Run)** del menú de **Inicio(Start),** se debe correr el comando Setup.exe.

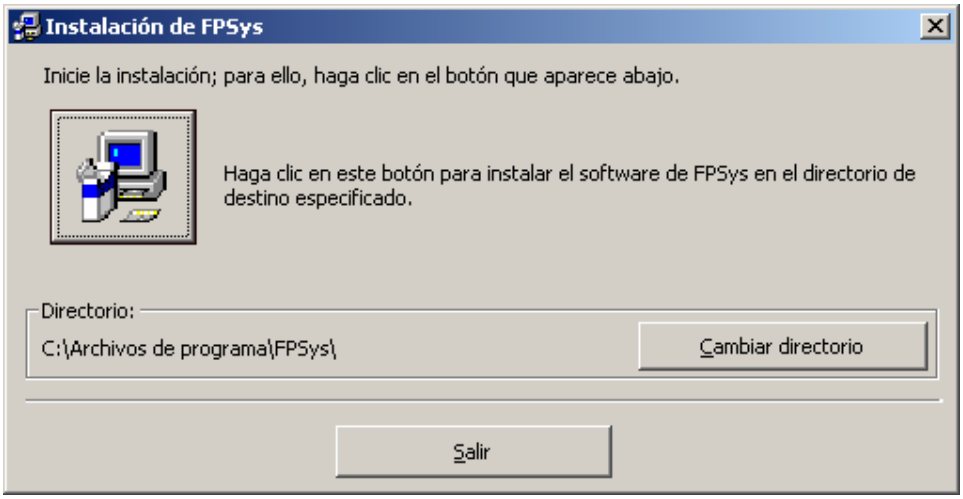


2. Los archivos requeridos por el sistema para su funcionamiento comienzan a ser copiados desde la fuente.

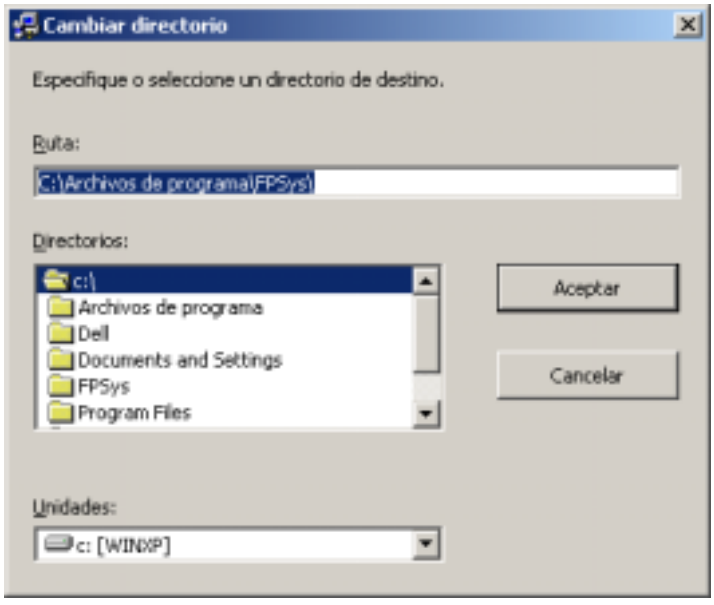
3. Se presenta la pantalla de Bienvenida. Se debe pulsar **Aceptar** para continuar con la instalación **o Salir** para cancelar la instalación.



4. En este paso de la instalación se puede cambiar el destino de los archivos de la aplicación con el botón **Cambiar Directorio.** De aceptar esta dirección se debe continuar con la instalación pulsando el icono de instalar.



5. Si se decidió a cambiar la dirección de destino de los archivos de la aplicación, con el botón **Cambiar Dirección,** debe ahora ingresarse la ruta deseada utilizando la siguiente pantalla. Luego debe continuarse con la instalación pulsando el botón **Aceptar.** De lo contrario se puede cancelar la operación de cambiar la ruta de destino con el botón **Cancelar.**

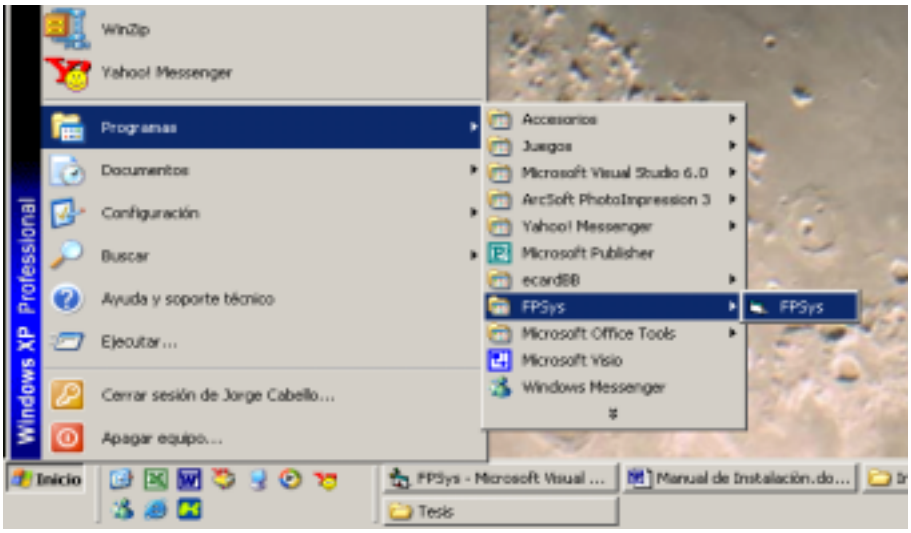


6. En este paso de la instalación se debe escoger el grupo de programas al que va a pertenecer la aplicación. Debe continuarse con la instalación pulsando el botón **Continuar.**

7. Luego de ingresar los parámetros requeridos comienzan a copiarse los archivos de la aplicación a la ruta especificada y los archivos de sistema requeridos. Se puede cancelar esta operación pulsando el botón **Cancelar.**

8. Al finalizar la instalación se presenta el mensaje de finalización exitosa.

Se puede ahora verificar la instalación y correr FPSys 1.0 desde **Programas(Program)** en el Menú de **Inicio(Start),** como se muestra en la siguiente figura:

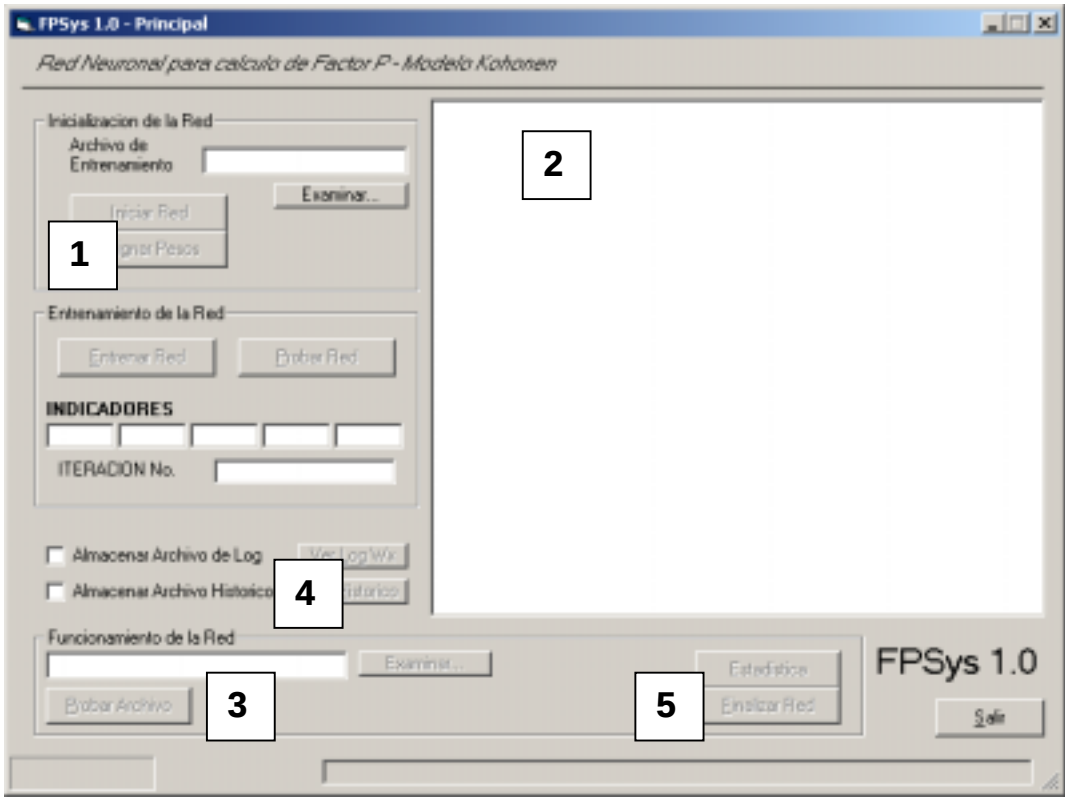


MANUAL DE USUARIO Y OPERACIÒN

INICIANDO LA RED

Antes de iniciar cualquier proceso en el sistema, se debe identificar correctamente las zonas de la pantalla para lograr el mayor control sobre la aplicación.

Así la pantalla principal posee 5 zonas importantes, las cuales pueden ser identificadas como se muestra en la figura.



De acuerdo a este gráfico cada una de las zonas representa:

Zona 1: Inicialización de la red. En esta zona se debe parametrizar la red e inicializarla.

Zona 2: Log de eventos (Visor de texto). Permite visualizar el estatus de la red, el proceso actual y eventos en la red.

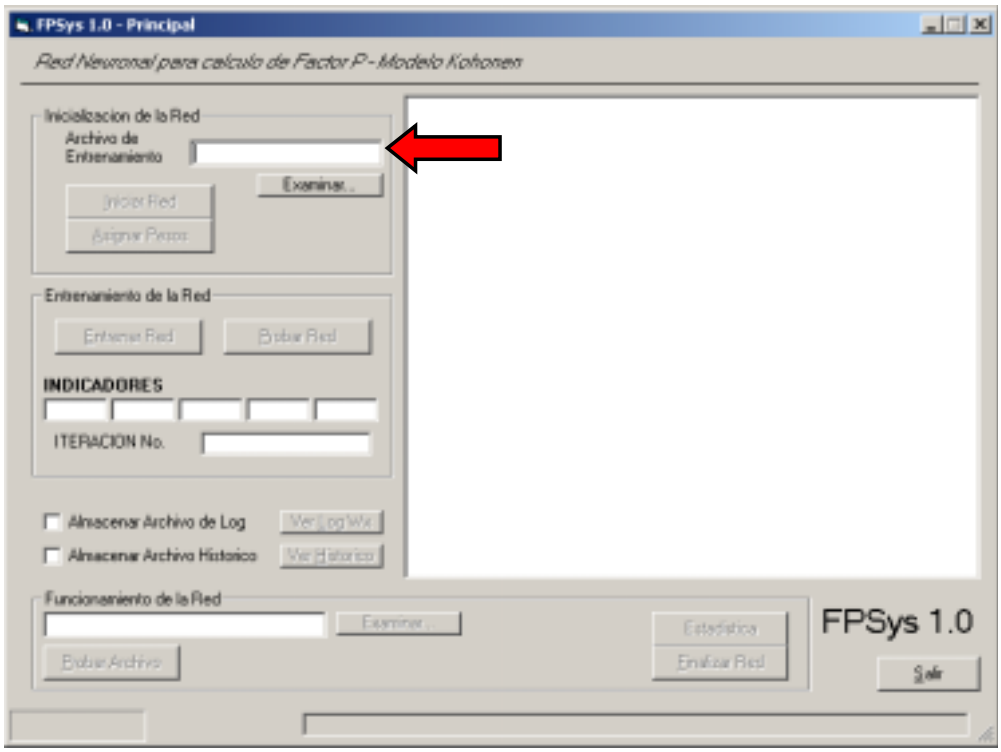
Zona 3: Pruebas en la red. En esta zona se puede hacer pruebas sobre la red ya entrenada.

Zona 4: Activación de logs e historial. En esta zona se puede habilitar los historiales y eventos.

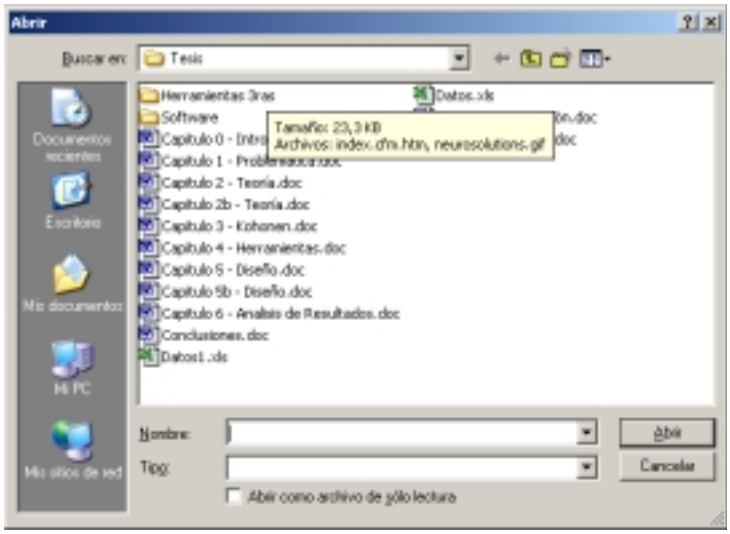
Zona 5: Ver estadísticas en la red y Finalizar la red.

Para iniciar un proceso de entrenamiento, previamente debe inicializarse la red con los parámetros correspondientes e ingresar datos con los que se va a entrenar la red. Con el propósito de dimensionar la red es necesario indicar la cantidad de datos a entrenar y el archivo que contiene los datos.

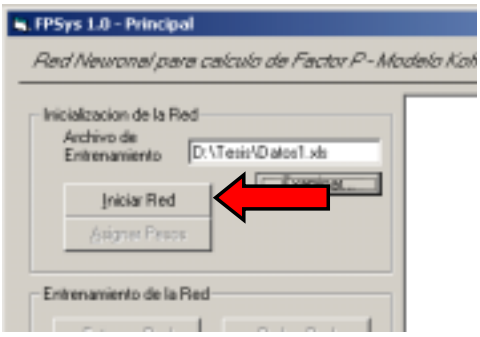
El botón examinar nos permite ingresar el nombre del archivo en Excel que contiene los datos que se usarán para el entrenamiento de la red.



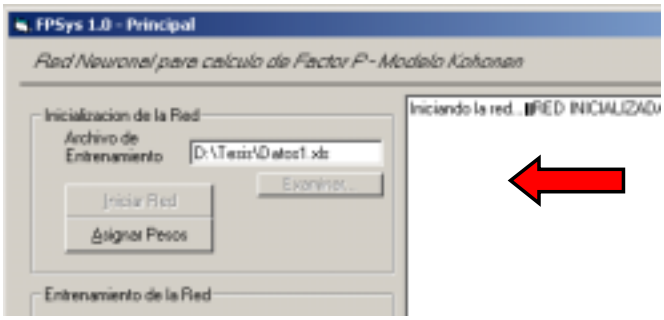
Se debe entonces escoger el archivo a utilizar para la inicialización de la red:



Una vez escogido el nombre del archivo, se debe observar el mismo en la pantalla principal.



Ahora podemos iniciar la red pulsando el botón de Iniciar Red. El visor de texto muestra el estado de la operación, tal como se ve en la figura:

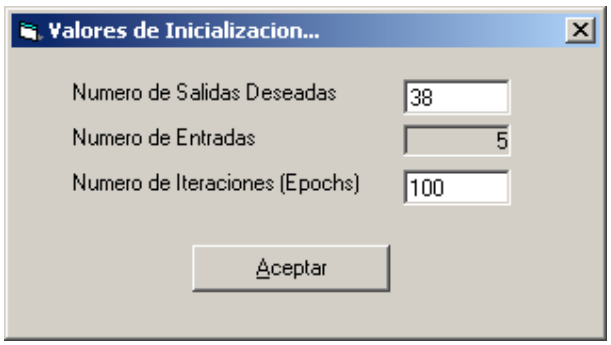


Luego de finalizada la etapa de iniciación de la red, se debe ingresar los parámetros de la red y ejecutar el proceso de Asignación de Pesos. Estos parámetros corresponden a:

***Número de Salidas Deseada**:* representa la cantidad de salidas que deben existir en la red.

***Número de Entradas**:* representa la cantidad de entradas en la red. En este caso este valor es fijo y de 5, dado que el número de variables para el cálculo del factor P es 5.

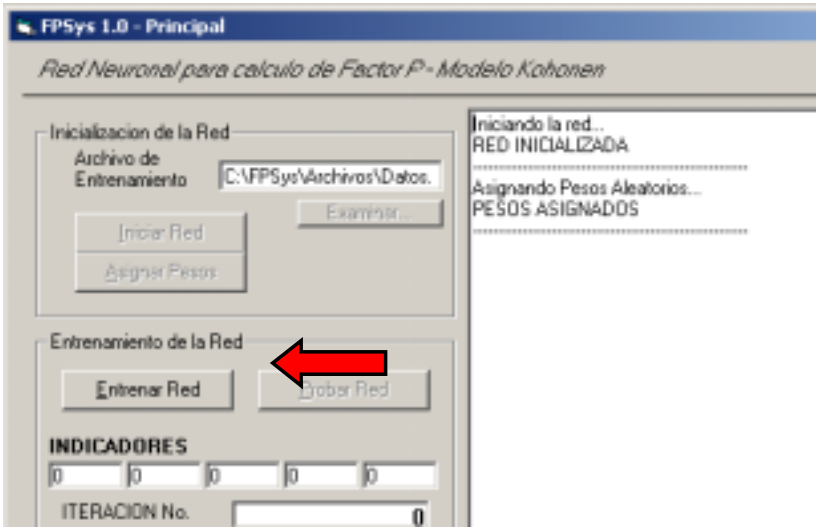
Número de iteraciones: representa la cantidad de veces que la red utiliza los datos en el proceso de entrenamiento.



ENTRENANDO LA RED

Una vez ingresada a la red los parámetros de ¡inicialización se puede entrenar la red. Antes de entrenar la red, se debe también asignar pesos aleatorios para cada uno de los datos que se han ingresado a la red. Esta operación se realiza pulsando el botón **Asignar Pesos**. El visor de texto indica el estado de esta asignación.

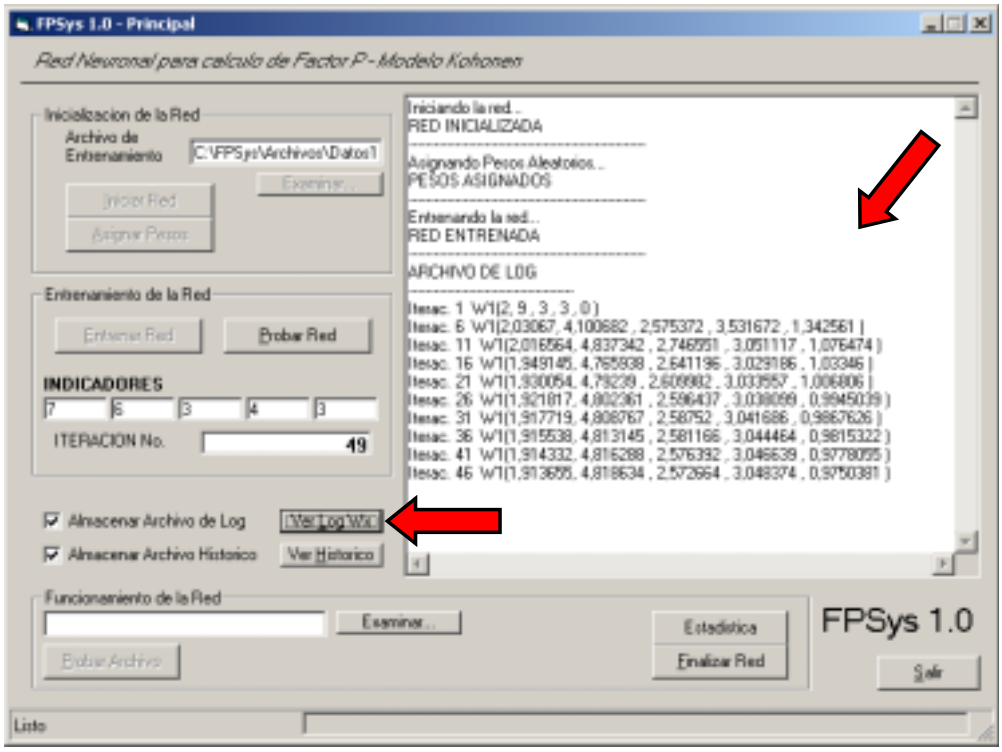
Luego de haber asignados los pesos aleatorios, se puede proceder a entrenar la red. Este proceso se ejecuta pulsando el botón **Entrenar Red**. El visor de texto indica el estado de la red, y visualiza el instante en el cual se finaliza la operación de aprendizaje. Adicionalmente, la barra de estado de la pantalla principal indica el porcentaje completado durante esta fase.



LOGS

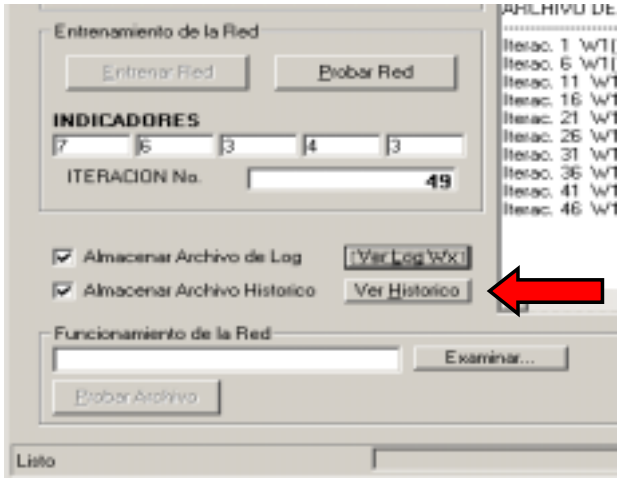
Existe la opción de habilitar dos tipos de historias o eventos en la red durante el proceso de entrenamiento: Un archivo de Log, que permite guardar un muestro del desarrollo de los pesos durante el entrenamiento; y un histórico que almacena el desarrollo completo de los pesos durante esta fase.

En el primer caso, si se habilita la opción Almacenar Archivo de Log el sistema crea un archivo llamado log.txt tipo texto en la carpeta c:\fpsys\logs. Este archivo se puede visualizar en el visor de texto de la pantalla principal pulsando el botón **Ver Log Wx**.



Este archivo almacena un muestreo de pesos de una neurona de la red escogida aleatoriamente. El muestro lo realiza para una sola neurona y lo hace cada 5 iteraciones durante el proceso de entrenamiento.

En el segundo caso, si se habilita la opción Almacenar Archivo Histórico el sistema crea un archivo llamado logfull.txt tipo texto en la carpeta c:\fpsys\logs. Este archivo no se puede visualizar en el visor de texto de la pantalla principal debido a que normalmente tiene un tamaño considerable. Este archivo se puede visualizar desde el visor de texto WordPad pulsando el botón Histórico.



Este archivo almacena el desarrollo completo de los pesos de todas las neuronas durante toda la fase de entrenamiento. Una muestra de este archivo se presenta a continuación:

Iteración No. 1
W 1 (5, 12, 4, 4, 0)
W 2 (2, 7, 4, 3, 0)
W 3 (1, 9, 1, 2 , 10)
W 4 (5, 5, 1, 0, 10)
W 5 (3, 4, 1, 4, 12)
W 6 (6, 6, 3 , 2 , 0)
W7 (3, 8, 2, 4, 12)

W 8 (6, 5 , 3 , 4, 2)

W 9 (5, 4, 3 , 2, 10)

W 10 (3, 1, 1, 2 , 6)

W 11 (2, 4 , 1 , 0 , 2)

W 12 (6, 10, 1 , 0, 14)

W 13 (6, 5, 1 , 4, 14)

W 14 (5, 11 , 3 , 4, 2)

W 15 (3, 6 , 4, 2, 14)

W 16 (3, 10, 1 , 0, 12)

W 17 (3, 7, 1, 3 ,12)

W 18 (2, 12, 2 , 0, 14)

W 19 (3, 3 , 3 , 2, 10)

W 20 (3, 4, 1 , 4,12)

W 21(1, 6, 2 , 4, 2)

W 22 (2, 1, 1 , 0 , 0)

W 23 (3, 5 , 2, 4, 12)

W 24 (2, 8, 2 , 4, 0)

W 25 (5, 1, 1 , 3, 14)

W 26 (6, 2, 1, 3, 10)

W 27 (2, 9, 3 , 0, 10)

W 28 (6, 1, 1 , 0 , 0)

W 29 (1, 8 , 4, 4, 6)

W 30 (1, 9, 2, 3, 10)

W 31 (3, 1 , 4, 2, 0)

W 32 (5, 11 , 4 , 0 , 2)

W 33 (3, 9, 1 , 2, 14)

W 34(6, 11, 4, 3, 12)

W 35 (1, 3 , 2, 0, 2)

W 36 (6, 11 , 4, 2, 14)

W 37 (3, 3, 1 , 2, 14)

W 38 (5, 5 , 2, 3, 10)

Iteración No. 3

W 1(2.179892, 1.445688, 1.588657, 3.806153, 1.22385)

W 2 (2.408267, 4.416888, 3.254404, 3.595706, 1.578462)

W 3 (5.844698, 4.698578, 3.058294, 2.905786, 10.11311)

W 4 (1.725032, 5.723288, 1.557905, 3.442381, 12.53899)

W 5 (5.281741, 11.1049, 2.543848, 2.954738, 6)

W 6 (4.212091, 8.600535, 3.474719, 1.06641, 1.000875)

W 7 (5.879785, 8.506929, 3.671996 , 2.313074, 10.00245)

W 8 (5.501511, 1.675323, 1.942947, 1.306152 , 6)

W 9 (2.354598, 4.018208, 2.034027, 1.094631, 13.78662)

W 10 (2.166803, 11.31544, 2.527669 , 3.692057 , 6)

W 11(1.873966, 1.953732 , 2.994957 , 6.251526E-02 , 6)

W 12 (5.794731, 2.496027, 1.190755, 3.218181, 11.18752)

W 13 (5.478559, 2.353332, 3.636482, 1.740311 , 0.758361)

W 14 (5.008728, 8.79471 , 2.861234, 3.837739, 1.968439)

W 15 (2.21133, 1.942219, 1.769915 , 0.3111556, 10.15678)

W 16 (1.347829, 6.65666, 1.199403, 1.03192, 10.31433)

W 17 (2.39268, 10.33695, 2.418874, 2.576692, 13.22601)

W 18 (4.000541, 6.616267, 3.19499 , 0.3697626, 12.9375)

W 19 (1.009659, 2.618663 , 2.359419, 3.882124, 11.50242)

W 20 (5.265533, 1.35979, 2.511541 , 8.593751E-02, 13.49096)

W 21(2.111917, 7.94556, 2.661002, 3.486104 , 6)

W 22 (1.202576, 3.67732, 3.062372 , 0.0123219, 0.8239412)

W 23 (5.809462, 11.10466, 1.697217, 5.587935E-09, 12.16909)

W 24 (2.080096, 11.34226, 1.650889, 2.332935 , 0.9349404)

W 25 (1.618689, 10.78994 , 3.48296, 1.461156, 10.50002)

W 26 (2.494711, 11.75493, 1.277935 , 0.5120522, 10.04512)

W 27 (1.761165, 2.550546, 3.966715, 2.285877, 11.50063)

W 28 (5.790112, 7.922604, 1.966237, 3.519291 , 6)

W 29 (2.425441, 2.705012, 3.428293, 2.725974 , 6)

W 30 (5.781229, 11.04171, 1.541098, 2.709593, 10.82177)

W 31 (1.440519, 8.617651 , 3.412571 , 2.709175, 1.745928)

W 32 (3.522236, 4.772148, 2.127427, 2.664665 , 6)

W 33 (1.646807, 1.891069, 2.31347, 2.186523, 13.9526)

W 34 (5.512025, 6.558255, 2.516866 , 3.92561E-03, 6)

W 35 (5.506083, 11.98375, 2.587899, 2.527568, 1.84492)
W 36 (1.726115, 7.76489 1 , 3.616029, 2.749469, 13.44125)
W 37 (5.292856, 5.705125, 1.714538, 3.332097, 12.17969)
W 38 (5.579229, 5.042253, 1.08191 , 2.233975, 1.007774)

Iteración No. 4
W 1(2.134456, 1.56009, 1.774851 , 3.576347, 1.133384)
W 2 (2.502009, 6.617685, 1.982103 , 3.332923, 1.137742)
W 3 (5.678872, 2.544805, 1.372199, 3.077068, 11.16109)
W 4 (2.122448, 4.211908, 1.963163, 1.631175, 13.56585)
W 5 (5.717784, 7.444552, 3.000494, 1.119678 , 6)
W 6 (1.674374, 9.786946, 2.070907, 2.076715, 1.998007)
W 7 (1.735772, 10.3784, 1.295418, 2.845146, 10.00014)
W 8 (5.538553, 2.176583, 2.433134, 0.4648866, 6)
W 9 (3.566845, 6.500383, 3.070628 , 9.08347E-03, 12.95203)
W 10 (2.079209, 10.7892, 2.584099, 3.404335 , 6)
W 11(1.849702, 2.089266, 2.755855 , 0.13 17295 , 6)
W 12 (5.797083, 2.791587, 2.648226 , 0.1271133, 11.30824)
W 13 (5.54983, 1.833374, 2.829753, 1.822833 , 0.5151041)
W 14 (5.431542, 10.11177, 3.2022, 0.5474588, 1.871585)
W 15 (2.535191, 1.071266, 2.343174, 0.6595765, 11.71325)
W 16 (1.314883, 5.453044, 1.327037, 1.122277, 10.01291)
W 17 (2.405026, 10.31979, 1.96245, 2.377433, 13.99999)
W 18 (5.835686, 8.581422, 3.416174, 1.828541, 10.23879)
W 19 (1.16902, 1.634839, 1.598073 , 3.02557, 10.47619)
W 20 (5.125527, 1.587551 , 2.373851 , 0.2149384, 13.9426)
W 21(2.401191, 6.612245, 2.035824, 3.358816 , 6)
W 22 (1.227335, 3.895415, 2.942293 , 0.6369849 , 0.590829)
W 23 (5.490883, 11.37461 , 2.077955, 1.040205, 12.85125)
W 24 (3.176964, 11.54429 , 3.083745, 3.299533, 1.039071)
W 25 (2.367047, 10.09651 , 2.854872 , 0.5887312, 10.8746)
W 26 (1.759205, 11.14763 , 2.752128, 3.184658, 11.7517)
W 27 (1.602682, 2.767414, 3.852026, 2.602739, 11.31036)
W 28 (5.436776, 9.845338, 2.059265 , 3.341469 , 6)

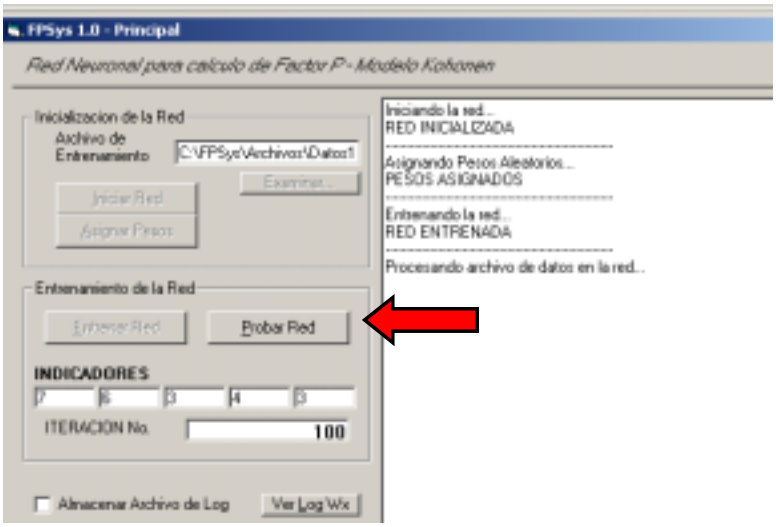
W 29 (3.32744, 2.23458, 2.873765, 3.227352 , 6)
W 30 (5.685139, 10.98304, 1.670481 , 2.970186, 10.93702)
W 31(2.812448, 8.318121 , 3.15592, 1.242298, 4.163607E-02)
W 32 (5.34822, 4.737648, 1.833709, 3.481879 , 6)
W 33 (1.77369, 1.895063, 2.282897, 2.576714, 13.69855)
W 34 (3.236776, 5.203324, 2.280383 , 0.7640234, 6)
W 35 (5.279729, 8.421657, 2.623858, 3.54079, 1.522046)
W 36 (1.82107, 7.701239, 3.565642 , 2.789124, 13.21838)
W 37 (5.521057, 5.349399, 2.11059, 3.218873, 11.30265)
W 38 (4.814376, 4.526591 , 3.204316, 2.935326,1.62044)

Iteración No. 5
W 1(2.077583, 1.616902, 1.83867, 3.394382, 1.112168)
W 2 (2.203801, 8.625784, 2.694289, 2.603219 , 0.9379028)
W 3 (5.623569, 2.539502, 1.518964, 3.004197, 11.14584)
W 4 (1.896345, 5.345075, 2.18593 1 , 4.500818E-02, 11.21641)
W 5 (5.515791, 8.323046, 2.471315, 3.34466, 6)
W 6 (1.998304, 11.32781, 2.594752, 3.169475, 1.127102)
W 7 (1.82238, 11.03817, 2.585363, 1.881506, 10.00038)
W 8 (5.483756, 2.290309, 2.429831 , 0.5840257 , 6)
W 9 (4.734603, 6.530698, 2.953807, 2.816188E-02, 13.99434)
W 10 (1.975308, 10.87679, 2.49802, 3.231315 , 6)
W 11(1.849524, 2.222557, 2.600264, 0.1714275 , 6)
W 12 (5.684993, 7.814803, 3.446249, 1.698756, 10.19032)
W 13 (5.743085, 2.12458, 3.115314, 3.385891, 1.114809)
W 14 (3.810871, 10.11831 , 2.276232 , 0.437882, 1.531133)
W 15 (2.467037, 1.296972 , 2.560379 , 0.6507593, 11.89159)
W 16 (1.913268, 6.476004, 1.728389 , 2.99486, 10.58152)
W 17 (1.749618, 9.091248, 2.795441 , 0.7749302, 12.58107)
W 18 (5.564686, 5.430177, 2.176567, 3.211039, 11.10284)
W 19 (1.75505, 2.799014, 2.968693 , 2.506756, 11.84688)
W 20 (5.319387, 1.634762, 2.383651 , 0.161755, 13.38584)
W 21(2.552863, 6.086482, 2.281421 , 2.987624, 6)

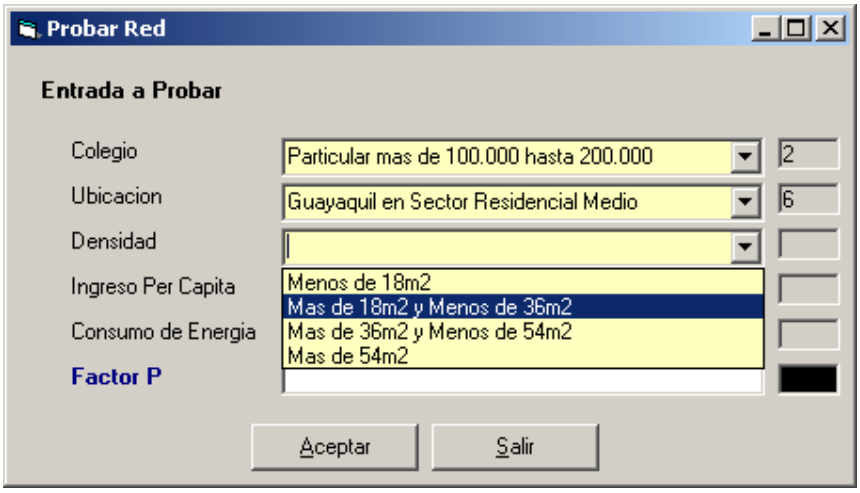
W 22 (1.896832, 4.229807, 3.032011 , 2.214203, 1.255472)
W 23 (4.09304, 10.60954, 2.586804, 2.039926, 13.99971)
W 24 (5.647146, 10.97631 , 2.726484, 2.891936, 1.711283)
W 25 (5.246765, 10.76036, 2.055066, 1.922224E-05, 11.27451)
W 26 (1.988956, 10.29639, 2.851592 , 3.573903, 11.99087)
W 27 (1.323792, 3.265995, 3.754097, 3.234033, 10.085)
W 28 (5.201018, 11.10219, 2.904974, 1.814043 , 6)
W 29 (3.083969, 2.384694, 2.836762 , 3.204265 , 6)
W 30 (5.658113, 10.96266, 1.745282, 3.078519, 11.00145)
W 31(2.393651, 7.766811 , 3.416611 , 0.6987328, 0.5564762)
W 32 (5.426891, 4.645889, 1.777316 , 3.332959 , 6)
W 33 (2.094599, 3.074299, 2.127409, 1.857086, 13.99988)
W 34 (4.613398, 6.45157, 2.627375, 7.373092E-02, 6)
W 35 (5.330439, 7.705525, 2.370747, 3.335133, 1.383015)
W 36 (1.621412, 6.369999 , 2.510501 , 3.055748, 13.97167)
W 37 (5.22009, 5.348283, 2.784809, 2.734074, 13.43764)
W 38 (5.307575, 3.479984 , 2.431755 , 0.8681723 , 0.1662636)

PROBANDO LA RED

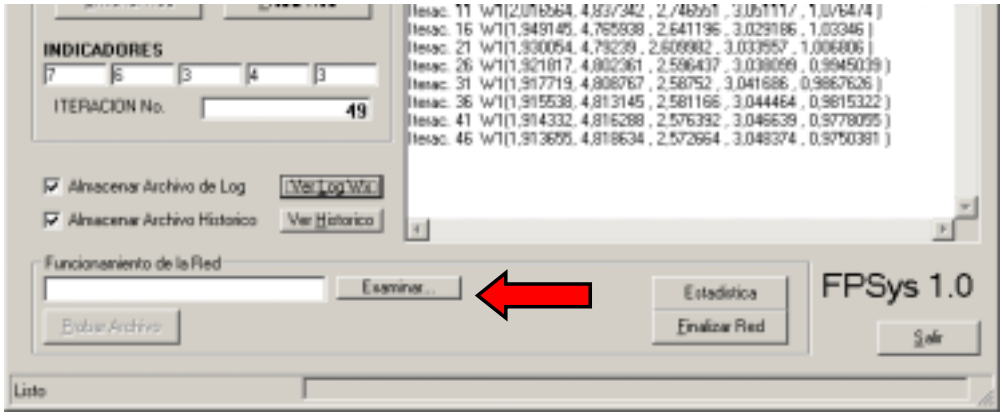
Luego de entrada la red se puede probar la misma de dos maneras. La primera opción es probar ingresando un nuevo conjunto de variables a la red y recibir la respuesta de la misma. En este caso sólo se puede ingresar una neurona nueva a la red a la vez. Esta opción requiere que se especifique los valores correspondientes al factor P, como son Colegio, Densidad habitacional, etc. Para probar esta opción se debe pulsar **Probar Red**.



En la siguiente pantalla se debe ingresar la información correspondiente al nuevo elemento que se desea ingresar a la red. Pulsando el botón Aceptar el sistema calcula el valor del factor P basándose en la red neuronal creado con los datos previamente ingresados.

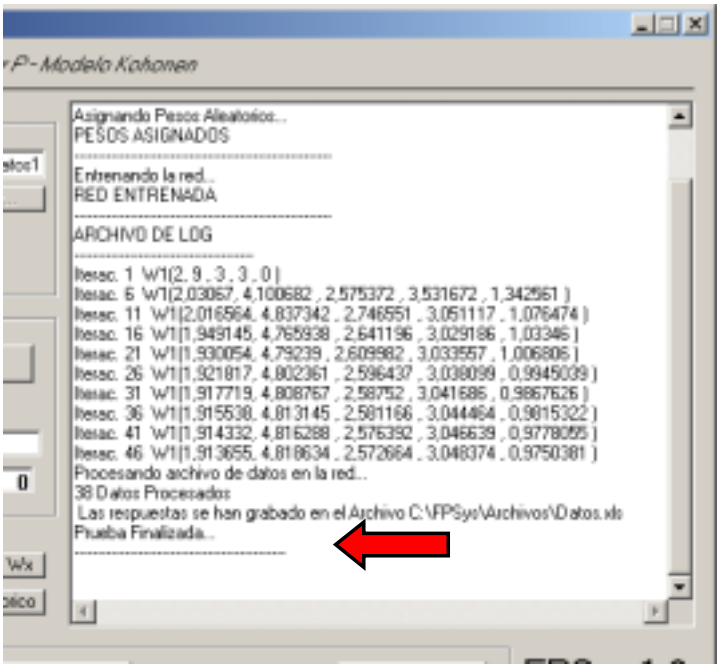


La segunda opción me permite ingresar más de una neurona a la red a la vez. Esto se hace a través de un archivo de Excel el cual contiene las nuevas neuronas que serán probadas en la red. El archivo a probar que contiene los nuevos datos debe ser especificado pulsando el botón Examinar.



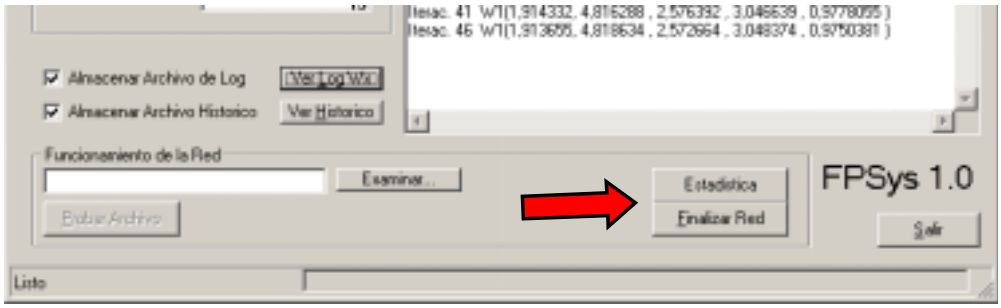
Pulsando el botón Probar Archivo se efectúa el proceso de prueba en la red y cada uno de los nuevos elementos procesados recibe su correspondiente factor P. Los resultados son almacenados en el mismo archivo de Excel el cual contiene los datos.

El visor de texto de la pantalla principal presenta el estado de la red durante este proceso, tal como se muestra en la siguiente figura:

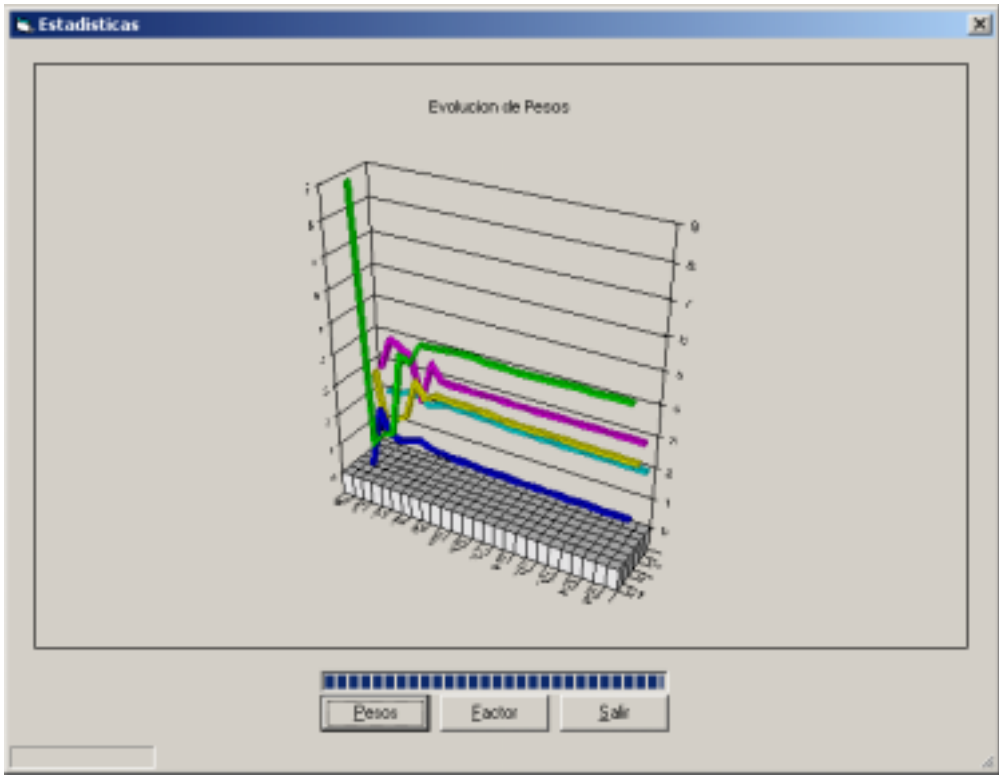


ESTADÍSTICAS

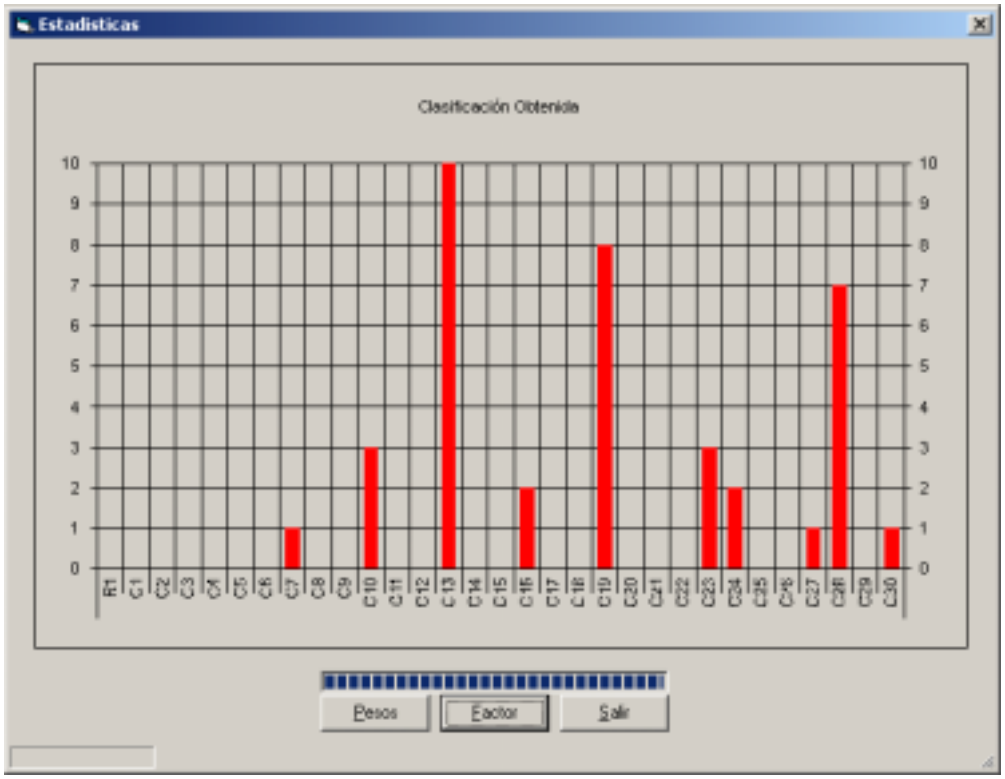
Se puede visualizar datos estadísticos capturados durante el proceso de entrenamiento y pruebas en la red. Se puede acceder a la pantalla de estadísticas pulsando el botón **Ver Estadísticas** en la pantalla principal.



Durante el proceso de entrenamiento el sistema lleva un control total sobre el desarrollo de los pesos para cada una de las variables de cada una de las neuronas en la red. De este modo se pueden llevar estadísticas por pesos. El sistema genera un gráfico de los pesos tal como se muestra en la siguiente figura:



Durante el proceso de prueba de un archivo para ingresar nuevas neuronas a la red, el sistema lleva un control sobre las asignaciones obtenidas para cada una de las neuronas. Esta clasificación se presenta de la siguiente manera:


Estadisticas
Clasificación Obtenida
10
9
8
7
6
5
4
3
2
1
0
R1
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
C20
C21
C22
C23
C24
C25
C26
C27
C28
C29
C30
Pesos
Factor
Salir